N O T I C E

THIS DOCUMENT HAS BEEN REPRODUCED FROM MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE

DESIGN OF A MODULAR DIGITAL COMPUTER SYSTEM

FINAL SUMMARY REPORT

CENTRAL CONTROL ELEMENT

AUTOMATICALLY RECONFIGURABLE MODULAR COMPUTER

One objective of the ARMS (Automatically Reconfigurable Modular System) spacecraft computer, developed by Hughes Aircraft Company for NASA, is to provide the capability to choose to maximize reliability through the use of redundancy and switchable spare modules or to maximize processing capacity by reconfiguration to provide multi-computing. Moreoever ARMS must be able to switch from one mode to another as a function of real time requirements, with no hardware changes, at a reasonable cost in power, weight, and volume. A CCE (Central Control Element) module to control this reconfiguration is the subject of this new technology disclosure. The CCE is a simplified implementation of the BOSS (Block Organizer and System Scheduler) module referred to but not described in this disclosure.

This logic has been implemented and breadboarded under NASA Contract NAS8-27926 for the George C. Marshall Space Flight Center, Huntsville, Alabama. It represents a "substantial advance in the state of the art" in that past computer designs have allowed redundant processing, or multi-computing but not both in the same computer with real-time mode switching. This new approach allows using the same hardware for either reliability enhancement, speed enhancement, or for a combination of both rather than for just one of these functions. This could prove very useful and cost-effective in a space mission or in other applications having some high reliability tasks and some other period of peak computation load during the computer's period of operation.

The ARMS computer controlled by the CCE consists of multiple memories and CPE's (Central Processing Elements), one or more IOP's (Input/Output Processors) and a Maintenance/Status Panel. These modules are standard computer building blocks with the exception of their interface logic as described in our previous New Technology Report. Each module contains internal detection logic utilizing redundancy and error detecting and correcting codes in keeping with standard techniques used in many modern computers.

(NASA-CR-161457) DESIGN OF A MODULAR DIGITAL COMPUTER SYSTEM Final Summary Report (Hughes Aircraft Co.) 48 p
HC A03/MF A01 CSCL 09B

N80-25009

Unclas
G3/60 21023

Thus CCE and interface logic concepts implemented in ARMS could also be applied to other general purpose computers needing ARMS attributes. ARMS modules can be configured for simplex, duplex, redundnat or triply modular redundant (TMR) operation.

M&S Computing, Inc., of Huntsville, Alabama, a subcontractor to Hughes Aircraft Company of this contract, was responsible for ARMS software development. No new technology was discovered in the course of this subcontract.

THE ARMMS COMPUTER

Any computer system justifies the cost of its development to the degree that it provides new capabilities or allows earlier ones to be satisfied at reduced cost. The Automatically Reconfigurable Modular Multiprocessor System (ARMMS) is primarily oriented toward providing the following new capabilities for spaceborne computers for application in the 1980 to 1985 time period.

1. To provide a modular computer system which is responsive to many mission types and phases.

2. To achieve through modularity a higher computing capability than previously available for spaceborn application. A target of several million instructions per second has been chosen.

3. To provide the capability to choose to maximize reliability through the use of redundancy or to maximize processing capacity through multiprocessing. This multi-mode capability must be dynamic; that is, a given system may alternate from one mode to another as a function of realtime requirements.

4. To maximize reliability in all applications through the incorporation of fault detection and recovery features and through the use of high reliability components.

The first consideration of any ARMMS design tradeoff is to avoid compromising these basic objectives. However, continuous concern must be maintained for the practical requirements of implementation.

ARMMS is an outgrowth and extension of two NASA development programs, the MSFC Space Ultrareliable Modular Computer (SUMC) and the ERC Modular Computer.

ARMMS consists of a grouping of Central Processor Elements (CPE's), I/O Processor (IOP's), Memory Modules, and a Block Organizer and System Scheduler (BOSS) module that will execute software routines for data and I/O scheduling, interrupt processing, system test, repair, and configuration, and power and clock switching and distribution. The IOP's and CPE's, and BOSS are connected to the memory modules by 4 pairs of buses as shown in Figure 1. One of the toughest challenges ARMMS faces is rapid reliable reconfiguration at a reasonable cost in power, volume, and complexity. A system of processor and memory interface logic that accomplishes this is the subject of this new technology disclosure.

INTERMODULE INTERFACE APPROACH

An intermodule interface has been designed that allows any CPE, IOP, or BOSS module to address any non-protected memory page. It allows any combination of simplex, duplex, or TMR streams with any combination of relative priorities to coexist with minimum bus contention, providing that no more than 4 CPE's, 4 IOP's, and BOSS are involved simultaneously. Volatile storage defining a module's role in ARMMS has been minimized and can be coded such that transients cannot cause an undetected change in the module's status. The interface allows all modules of a class (CPE, Memory, etc.) to be virtually identical. Interface gate complexity and module-to-module interconnections have been minimized.

Whenever a stream is formed, BOSS sends each processor module involved a stream status code defining all bus connections within the stream and that stream's priority. Once assigned to a stream, a processor always uses the pair of buses specified by the stream status code for communication to and from memory, eliminating bus contention among processors of a given type. For redundancy, each processor can output on a choice of two buses. This choice is made by BOSS command. To reduce bus contention between processors of different types, a hierarchy is established such that I/O and BOSS modules can inhibit CPE modules from starting a new memory access cycle when the

former modules require access to a memory bus. Similarly, BOSS (but not CPE) modules can inhibit I/O modules' bus access. Once any module has been granted access, it will continue to have it until transfer of the word involved has been completed. Usually, only processors using buses needed by other processors are inhibited, except that all processors operating synchronously in a duplex or TMR stream are inhibited if one or more processors in the stream are inhibited ensuring maintenance of synchronization between these processors. Modeling indicates that speed lost due to bus contention between processors of different types should be less than 3% exclusive of memory contention losses that are independent of the interface design.

BOSS assigns each memory module a page address and a high, middle, or low bus response assignment in case of memory accessible by a TMR stream (or a high or low assignment for access by a duplex stream). Memory page size will equal memory module size. All memory modules assigned to a given page output on the same bus when accessed by a simplex stream or on different buses according to their bus response assignment when accessed by duplex or TMR streams. Examples are shown in Figures 2 and 3. All duplex or TMR stream processors receive memory outputs on all buses assigned to that stream. Each processor access request contains a page address and a bus priority code. Processors will continue to request access until it is granted or until they are temporarily inhibited by other processor's desire to access.

The assignment codes discussed above require 6 bits from BOSS to memories, and 9 bits from BOSS to processors, plus extra bits for error detection coding. Each module input interface includes voting and fault detection coding logic. These interfaces can be implemented at an estimated complexity of 1000 gates per module.

The ARMMS priority structure will involve both hardware and software elements. The hardware recognizes a minimum of 16 different priority levels. The software then selects different subsets of these 16 as program requirements dictate. The highest hardware priority goes to BOSS, since the efficiency of the rest of the system depends on BOSS completing its tasks efficiently. The second highest priority is a special TMR CPE mode used only in the event of an error in one of three TMR channels to ensure completion of the TMR task with maximum speed prior to initiating diagnostic tests on the stream. The next seven priorities are for I/O streams on the assumption that the timing of external events happening and mass data transfers is more difficult to control

than the timing within processing streams and, hence, IOP memory access requests should be given higher priorities than CPE access requests. The seven lowest priorities are for CPE's. Different numbers of arrangements of priorities could be easily implemented if required.

So long as BOSS, I/O, and CPE programs are mostly segregated into different memory pages, all 3 types or programs should be able to be executed simultaneously with minimal bus or memory contention. When these programs wish to access the same memory page, the internal logic design of the memory access logic will tend toward letting the streams access the memory a word at a time in turn, since each processor will release the memory temporarily between access requests, letting the next higher priority stream gain access for one word. This results in all contending streams slowing down, but none stopping entirely. Obviously, this does not preclude the need for designing the software to minimize memory contention if ARMMS is to perform efficiently as a multiprocessor.

The seven priority levels available for normal I/O and CPE scheduling are ordered in descending priority as shown in Table I, allowing the 14 modes listed in the table. The logic allows any of the combinations listed for CPE's to be used simultaneously with any of the combinations listed for IOP's. Note that the choices allow for any combination of relative priorities between streams of differing criticality, and that the software system can change the priority assignment of a given stream at will; also, that combinations such as 2 duplex IO streams and a simplex plus a TMR processing stream are allowed. If IOP's and CPE's are to be tied together in the concept of "full processing stream" via software, both processor types could be given either the same CPE or the same IOP priority assignment by BOSS. Otherwise, BOSS assigns IOP's only I/O priority codes and CPE's only CPE priority codes, and the hardware provides for complete independence of the I/O and processing streams subject only to software restrictions.

In order to access data from memory, a processor must provide a 4-bit page address to select one of 16 memory pages, a 4-bit priority request to allow the given memory page to choose the highest priority stream's request and determine if the correct number of processors agreed on this request, the number being determined by the priority's mode (simplex, duplex, or TMR), a 3-bit 2 out of 3 coded Read/Write/Transfer request, and a 13-bit word address to select one of up to 8,192 words in a memory module. The first 8 of these 24 bits must be

present for a memory to make a decision as to whether or not to grant the request. In addition, a sync or "access request" signal must be present to tell the memory that it is supposed to be making such a determination if these 8 bits are to be transmitted on lines that can also carry word addresses and data that might otherwise be confused with page and priority information. The processor to memory bus must be at least 8 bits wide plus the access request line and any desired parity lines in order to function efficiently.

In addition to data lines, if the buses are less than a full word wide the memory to processor bus must contain a dedicated memory response line to signal the processor that the first bits of address have been accepted and the processor is to continue the transmission to completion. If a processor does not receive this response signal, it will continue to transfer the first bits of the address to the memory interface until either the processor is inhibited by another processor or the memory responds to the data. Since only one processor can use the bus at a given time, all requests and responses are unambiguous.

Three additional lines are required in connection with the memory buses at the processors only. Each processor receives inhibit lines from each of the other two classes of processors and sends an inhibit to these other two classes, describing each processor's bus activity. In addition, an I/O busy line may be required from IOP to CPE in the event of several CPE's wishing to access a given IOP simultaneously. This will depend on the details of the IOP's and is shown for completeness. Note that the BOSS module receives the IOP's Memory Access Request as an inhibit rather than the IOP's normal inhibit line which does go to the CPE. This is because the IOP's memory access request line will not go true until all buses needed by the IOP have been cleared of traffic and, hence, this line will inhibit BOSS only in the event that the IOP can gain access to the memory through use of free buses or inhibiting CPE's, maintaining BOSS priority over the IOP. The information to be transferred to or from a memory by processors is summarized in Figure 4, assuming a 32-bit data word plus 7 error correction code bits and a 13-bit bus width.

INTERFACE LOGIC DETAILED DESIGN

Within each processor (BOSS, CPE, or IOP) is an access request network that will request memory access whenever an appropriate bit appears in the

processor's microprogram, subject to the inhibitions (BOSSINH, etc.) from other processors. Figure 5 shows a gate level drawing for this logic in the case of the CPE module. Logic for IOP and BOSS is similar and is shown in Figures 6 and 7. The choice of inhibiting factors is controlled by the Stream Assignment Register in the CPE or by hardware connections in BOSS and IOP, with BOSS having highest priority to memory, IOP middle priority, and CPE lowest priority. The logic also correlates memory responses (MEMRES) to its access request (MEMREQ) and, when a response from the correct memory modules occurs, sets a flip-flop (AGF) allowing the access to go to completion and inhibiting other access to the bus until the cycle is complete as signaled by a second microprogram bit within the processor. IOP and BOSS access control logic differs from that of the CPE only in that an Access Request Flip-Flop is incorporated (ARF) which locks out lower priority modules from accessing memory while these higher priority modules are requesting memory access. All modules can lock out others while they are actually accessing memory instead of merely requesting it.

Figure 8 gives a detailed view of the logic within each memory module's access control block. Figures 9 and 10 show the same logic at a gate level. As the data comes in on each bus, buses whose access request lines are true and have page addresses agreeing with a memory module's page address (PGID) will be tested for access to the memory registers. The 16 priorities (A_i..Pi) are decoded and applied to the request detection and priority ordering logic. If this circuit detects the correct number of requests of the highest priority present at the time of the test (BOSS...CPE SMPLXD) and the memory is not already in use (DS_1...DS4=0), the memory responds (RS_1=MEM RES) on the buses assigned to the processor generating the request and gates the response decision into the Response and Criticality fields of the Assignment Holding Register and to the voting logic to allow the voted data to go to the memory registers and to set up the proper output bus paths for the memories' data input in the case of a Read. When the cycle is complete,the Response and Criticality fields of the Assignment Holding Register are cleared, and the memory is ready for the next access. The bus output mode field determines which of 3 TMR buses a memory module will output in TMR according to an assignment from BOSS.

Each module contains voting logic which will vote any combination of 3, compare any combination of 2, or transfer any one bus's inputs to an appropriate

module register, signaling any disagreements to the module's status/command network which will interrupt BOSS as appropriate. In processor modules, the voter paths are controlled by the Stream Assignment Register, while in memory modules they are under the control of the Response and Criticality fields of the Memory Assignment Holding Register. This logic allows for maximum software flexibility in the ARMMS configuration process with a moderate amount of hardware.

Figure 11 shows 3 simple circuits for interfacing with bused data. The first allows masking of "stuck on 0" failures in the duplex and TMR modes on the assumption that the transmitting module was designed to transmit "0" when it had detected an internal failure. This circuit could then be followed by error detection or correction logic. This circuit also allows straight-through transmission of simplex data. The second circuit is a basic voter for use in TMR only. It does not allow error detection; only correction. The first and second circuits could be used together for a full simplex, duplex, TMR capability. The last circuit provides a fault detection add-on, for TMR only, that signals a fault when no combination of 3 bus inputs agree. The principal advantage in this circuit is that while it detects faults, it does not say which bus was at fault.

Figure 12 shows the voter/switch used in baseline ARMMS. It incorporates all the features of the three circuits discussed above, plus allowing fault isolation to a specific bus. This circuit normally allows ORing together any enabled bus signals as in the first circuit above. Simultaneously, it votes on the enabled (DS_i) data inputs in TMR and generates a fault signal (FLT_i) for any enabled bus input that disagrees with other enabled inputs. This fault signal is output to the module's fault control logic and is used to prevent that bus's data from passing through the data-ORing section of the voter switch.

The intermodule interface circuits described have a gate delay of 17, including 5 in the voter switch, 2 in the processor access control logic, and 10 in the memory access control logic. This amounts to a 51 nsec propagation delay, assuming a 3 nsec average delay per gate for LSI silicon-on-sapphire CMOS logic. For a 10 MHz data bus transfer clock rate, this would leave 49 nsec for bus driver, receiver and transmission delays.

Central Control Element (CCE). The Central Control Element distributes power and clock signals to other ARMS modules and coordinates ARMS reconfiguration either due to new assignments from the maintenance/status panel or in response to fault interrupts from other ARMS modules. In order to minimize costs the breadboard CCE does not include redundancy that could be implemented in a flight version. For maximum reliability a TMR CCE with voting between the parts on all outputs would be desirable.

The CCE consists of individual status controllers for each ARMS module to be controlled fault correlation logic, an overall program initiator and reconfiguration controller, switching logic for power supplied to other modules, a crystal controlled central clock source, and external interrupt routing logic. The CCE has no internal processing or main memory bus access capabilities but is capable of utilizing CPE software or hardware to enhance its own hardwired capabilities by means of interrupts. A block diagram of the CCE is shown in Figure 13. The following is a description of the specific enbodiment of the CCE used in the ARMS breadboard:

CPE Module Status Controller. One CPE module status controller is required for each of the 4 CPE modules in the ARMS breadboard. Each controller keeps track of the CPE's status (spare, active normal, active abnormal, failed) outputting a stream assignment bit corresponding to that CPE's hardwired processor (to memory) bus. Together the 4 CPE module status controllers provide a 12 bit stream assignment to all CPE's identifying which CPE's are active and which are passive. When the CCE is powered initially, each CPE module status controller places its CPE in the spare state. A signal from the maintenance/status panel causes one or more of these controllers to place their CPE's in the "active normal" state. If a fault interrupt from either a CPE or a memory module indicates that a specific CPE may have failed that CPE's status controller is placed in the "active abnormal" state. Figure 14 shows the various states that a module status controller may take on.

If the CPE is operating in the simplex mode when the fault was detected, or if it is operating in the duplex mode and the fault is detected by a memory module without being internally detected within the CPE, the CPE module status controller causes the Program Initiator and Reconfiguration Controller (PIRC) logic discussed in the next section to issue a stop CPE interrupt immediately. If the CPE is operating in the TMR mode when the fault was detected, or if it

is operating in the duplex mode and the fault is internally detected within the CPE, the controller issues a stop CPE interrupt immediately following a receipt of a CPE available/rollback pace signal from the CPE, or after a prescribed time interval, whichever is shorter. Once in the "active abnormal" state one of the following events occurs in the CPE module status controller:

(a) If the CPE issues a CPE available/rollback pace signal prior to receipt of another fault interrupt concerning this CPE the status controller returns the CPE to the "active normal" state.

(b) If another fault interrupt concerning the CPE is received prior to receipt of the CPE's available/rollback pace signal the controller enters the failure pending state. From this state the reconfiguration controller either replaces the faulty module if it has sufficient priority and a spare is available, transferring its assignment to the spare CPE and causing the CPE module status controller to place its CPE in the failed state, or otherwise the reconfiguration controller returns the CPE module status controller to the active abnormal state. Thus modules that cannot be immediately replaced continue to be retried, and ARMS continues to operate in the presence of maskable failures.

Fault interrupts from IOP or main memory modules cause issuance of a stop CPE interrupt immediately if the CPE is operating in the simplex mode or immediately following receipt of a CPE available/rollback pace signal from the CPE if the CPE is operating in the duplex or TMR mode. The CPE module status controller remains in the active normal state during this operation in the absence of a fault interrupt placing blame on the CPE. The CPE module status controller also issues stop CPE interrupts prior to any external command update of assignments from the maintenance/status panel or due to an emergency such as an impending power failure.

IOP Module Status Controller. The IOP module status controller design requirements are similar to those for the CPE module status controller with the following exceptions:

(a) A stop IOP interrupt will not be issued unless the IOP does not stop within a prescribed time interval after all CPEs have halted.

(b) An IOP will not be returned to the "active normal" state from the

"active abnormal" state unless all active CPEs issue CPE available/ rollback pace signals prior to the receipt of another fault interrupt concerning this IOP.

Main Memory Module Status Controller. One main memory module status controller will be required for each of the 4 main memory modules in the ARMS breadboard. These controller's design requirements will be similar to those for the CPE module status controller with the following exceptions:

(a) A stop memory interrupt is not required.

(b) A main memory module will not require stream assignment status bits but will require page address and output bus assignments. The output bus assignment determines if a memory module will transfer data to CPEs or IOPs on the lower, (middle), or upper numbered memory (to processor) bus paired with the processor (to memory) buses to which access was granted. An "essential/non-essential" memory status bit is also required internal to the main memory module status controller to determine the proper memory replacement algorithm for the reconfiguration controller in response to a memory fault interrupt. An essential memory contains programs and important data the loss of which could disable a stream. A non-essential memory contains working storage and other contents the loss of which would not disable a stream.

(c) A main memory will not be returned to the "active normal" state from the "active abnormal" state unless all active CPEs issue CPE available/rollback pace signals prior to the receipt of another fault interrupt from this memory.

Program Initiator and Reconfiguration Controller. The program initiator and reconfiguration controller (PIRC) restarts the ARMS CPEs initially, or if they have been stopped for any reason, and controls the transfer of status assignments between individual module status controllers when ARMS reconfiguration is required.

The program initiator logic is activated whenever a load request is received from the maintenance/status panel, any faults are detected, or CPE available/rollback pace signals are not received from all CPEs within an

interval timed by the PIRC logic. The various states that the PIRC logic can assume are shown in Figure 15. Once activated, the program initiator logic issues stop interrupts to the CPEs as discussed in the previous section, issues a panic halt signal to all CPEs and IOP, waits for CPE available/rollback pace signals from all CPEs and an IOP available signal to stabilize in the available states, and then takes one of the following actions in descending priority:

(a) In the case of an essential memory failure in the duplex or TMR mode the program initiator logic issues a clear memory interrupt to the questionable memory, forcing its output to "0" pending completion of initialization, followed by an initialize memory interrupt, along with control information specifying the memory page to be initialized, to the highest priority CPEs. These CPEs enter a program that alternately reads from and then writes into every word in that memory page duplicating data from the good memory(s) into the newly assigned memory. All zero output conditions from the memory being initialized shall be considered to be normal until this operation is completed as signaled by a rollback pace signal from the CPE in question. Upon receipt of this signal the program initiator logic issues start interrupts to any remaining active CPEs if more than one processing stream is used in ARMS and restores the newly initialized memory to normal operation. Upon completion the memory initialization program automatically returns to the appropriate rollback point of the program in progress at the time of the interrupt.

(b) In the case of any other failure the program initiator logic issues start CPE interrupts to all active CPEs causing them to return to the appropriate rollback point(s) for the program(s) in progress at the time of the interrupt.

Figure 16 shows the PIRC logic necessary to respond to CPE rollback pace signals and to issue the interrupts discussed above. The reconfiguration controller controls the transfer of status assignments between individual module status controllers in response to commands from the breadboard's maintenance/status panel or to any of the individual module status controllers entering the failure pending state. Transfers of status assignments from failed active modules to newly activated spare modules occur once the program initiator logic verifies that the IOP and all CPEs are available (i.e., stopped) and prior to issurance of any interrupts by the program initiator logic with the following restrictions:

(a) Only one module of any given type can be replaced at a time and a spare module of that type must be available. For example, one memory plus one CPE may be replaced but not two CPEs at one time. If two CPEs did fail at once, one would be retried a second time and if it still malfunctioned and an additional spare CPE was available it would then be replaced.

(b) Essential main memory modules operating in simplex cannot be replaced by spares since no mechanism for initializing them is available. A permanent failure in such a memory module requires outside intervention for correction.

The logic for transferring assignments between status controllers is shown in Figure 17.

Fault Correlation Logic. The fault correlation logic allows the CCE to maximize the probability of correctly isolating a fault to a specific ARMS module within limitations dictated by a reasonable level of hardwired logic complexity and allows the CCE to determine that certain faults are maskable so that critical programs can continue to completion. The CCE correlates received fault interrupts from each CPE, IOP, and main memory module with appropriate status information from their status controllers as shown in Figure 18.

Many CPE and IOP faults may be isolated due to fault interrupts from the module in question. Single memory module fault interrupts indicate failures within the interrupting memory. In duplex and TMR modes simultaneous fault interrupts from two or more memories can isolate a failure to a CPE or IOP module whose identify is encoded in the interrupt. In the duplex mode these interrupts may only isolate the fault to one of two CPEs or IOPs in the absence of a direct fault interrupt from the offending module. However, an arbitrary replacement of one of these modules provides 50% probability of success in cases that otherwise would result in an ARMS system failure.

In simplex mode detectable faults (other than maskable single bit failures within main memory modules) result in immediate rollback or replacement of the offending module. In the absence of a fault interrupt from the CPE or IOP the fault is blamed on non-essential memories or on the CPE or IOP accessing an essential memory in the case of an ambiguous fault. If a fault is unambiguously isolatable to an essential memory the fault is insolvable since no mechanism

exists for initializing a spare in this mode. Some faults may be undetectable in simplex mode.

In duplex mode virtually all faults are detectable and at least those detectable in simplex allow the program to continue to its next rollback point and then are correctable in real-time through reconfiguration so long as spare modules are available. In all modes ARMS breadboard is capable of continued computation in the presence of faults so long as these faults are maskable. The choice between rollback and continued computation is software determined in that it is dependent upon whether the program is stopped before or after the program status block is updated. If the block has been updated the next program is executed, if not, then the present program is repeated. Programs shall be constructed so that they can be repeated if necessary.

Power Switching Logic. The CCE distributes power to all other ARMS modules. The power switching logic provides power to each ARMS module whose individual status controller places it in either an "active normal", "active abnormal", or "failure pending" state.

Crystal Controlled Block. The CCE contains a crystal controlled oscillator providing central clock signals to all ARMS modules to assure their synchronization.

External Interrupt Logic. The CCE holds external interrupts when they are received and routes them to the CPEs for which they were intended. When a CPE responds to a given interrupt it sends a response to the CCE which clears the interrupt once it receives response from a majority of the CPEs to which the interrupt was sent. As in the case of the power and clock distribution external interrupts are routed through the CCE since it is the only element in ARMS which remains stable throughout system reconfiguration. Clock Distribution and External interrupt logic is shown in Figure 19.

CCE Technology and Component Count. A CCE has been breadboarded out of T^2L small scale integrated circuit logic. For maximum reliability it should ultimately be implemented with CMOS LSI technology. Table 2 shows the number of gates and flip-flops required by each part of the CCE. Clearly the CCE complexity would increase for larger numbers of controlled modules but for ARMS it contains less than 1200 equivalent gates and is simple enough to be readily implemented on 2 or 3 large scale integrated circuits if desired.

TABLE I. ARMMS PROCESSOR PRIORITY ASSIGNMENTS

		Priority Code	Proc. Type	Stream Criticality	
1.	(Highest)	0000	BOSS	TMR	
2.		0001	CPE	TMR (Special)	
3.		0010	IO	SIMPLEX A	(SA)
4.		0100	IO	DUPLEX A	(DA)
5.		0110	IO	TMR	(TR)
6.		1000	IO	SIMPLEX B	(SB)
7.		1010	IO	DUPLEX B	(DB)
8.		1100	IO	SIMPLEX C	(SC)
9.		1110	IO	SIMPLEX D	(SD)
10.		0011	CPE	SIMPLEX A	(SA)
11.		0101	CPE	DUPLEX B	(DB)
12.		0111	CPE	TMR (Normal)	(TR)
13.		1001	CPE	SIMPLEX B	(SB)
14.		1011	CPE	DUPLEX B	(DB)
15.		1101	CPE	SIMPLEX C	(SC)
16.	(Lowest)	1111	CPE	SIMPLEX D	(SD)

NOTE: IN A FULL PROCESSING STREAM AN IOP MAY BE GIVEN THE STREAM'S CPE PRIORITY CODE.

IOP AND CPE STREAMS MAY INDEPENDENTLY HAVE THESE 14 MODES:

4 Processors	3 Processors	2 Processors
(SA, TR) or (TR, SB)	(TR)	(DA)
(DA, DB)	(SA, DA) or (DA, SB)	(SA, SB)
(SA, SB, DB) or	(SA,..., SC)	
(SA, DA, SB) or		1 Processor
(DA, SB, SC)		
(SA, ..., SD)		(SA)

TABLE 2

CCE COMPONENT COUNT

Function	Gates	Flip/Flops	Total Equiv. Gate
1. CPE Status Controller	131	28	299
2. IOP Status Controller	51	8	99
3. Memory Status Controller	159	36	375
4. Program Initiator/ Reconfiguration Control	119	13	197
5. Fault Correlation	134	0	134
6. Clock Control/Distribution	14	4	38
7. Ext. Interrupt Logic	20	2	32
	628	91	1,174

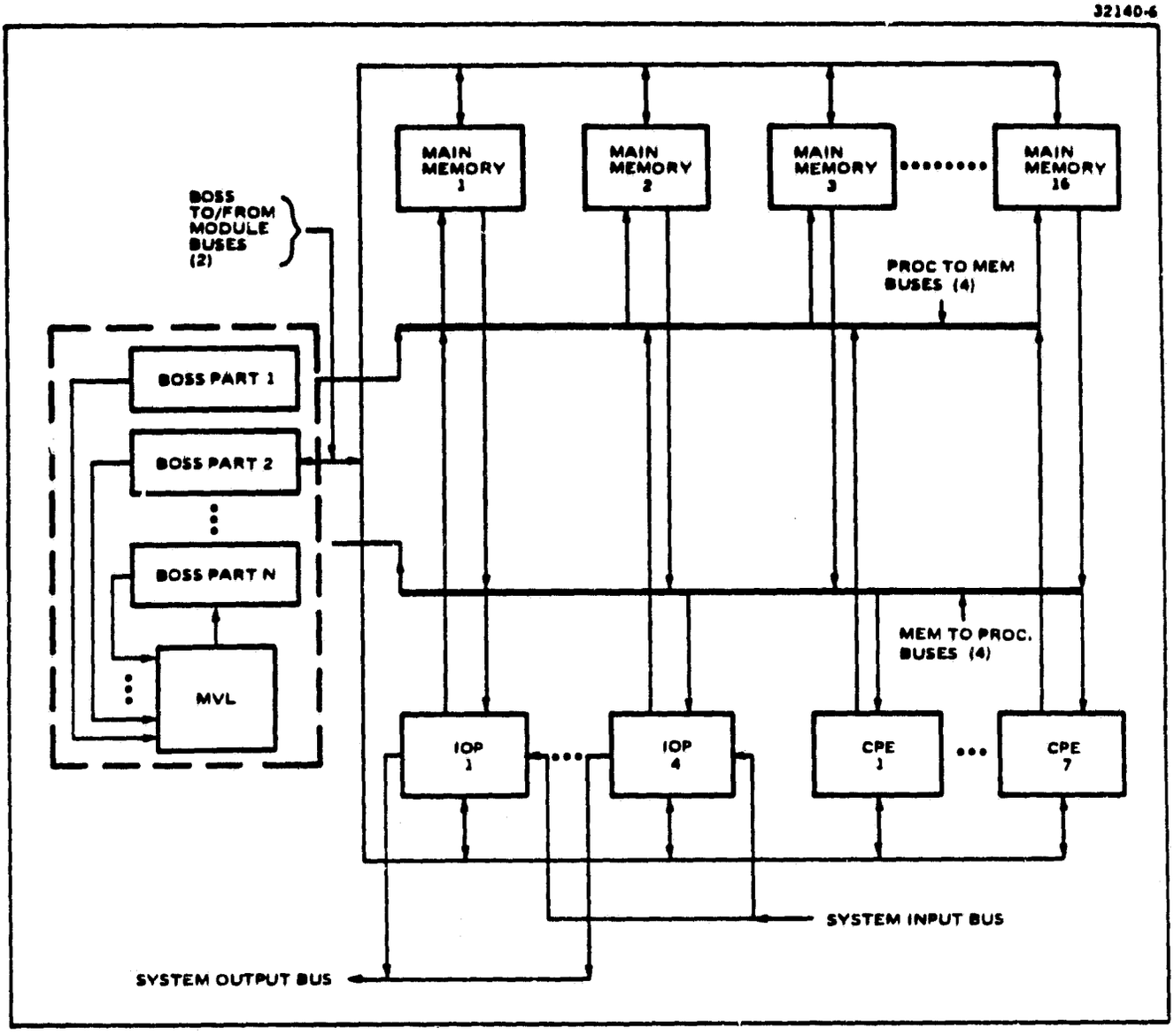


Figure 1. ARMMS System Configuration

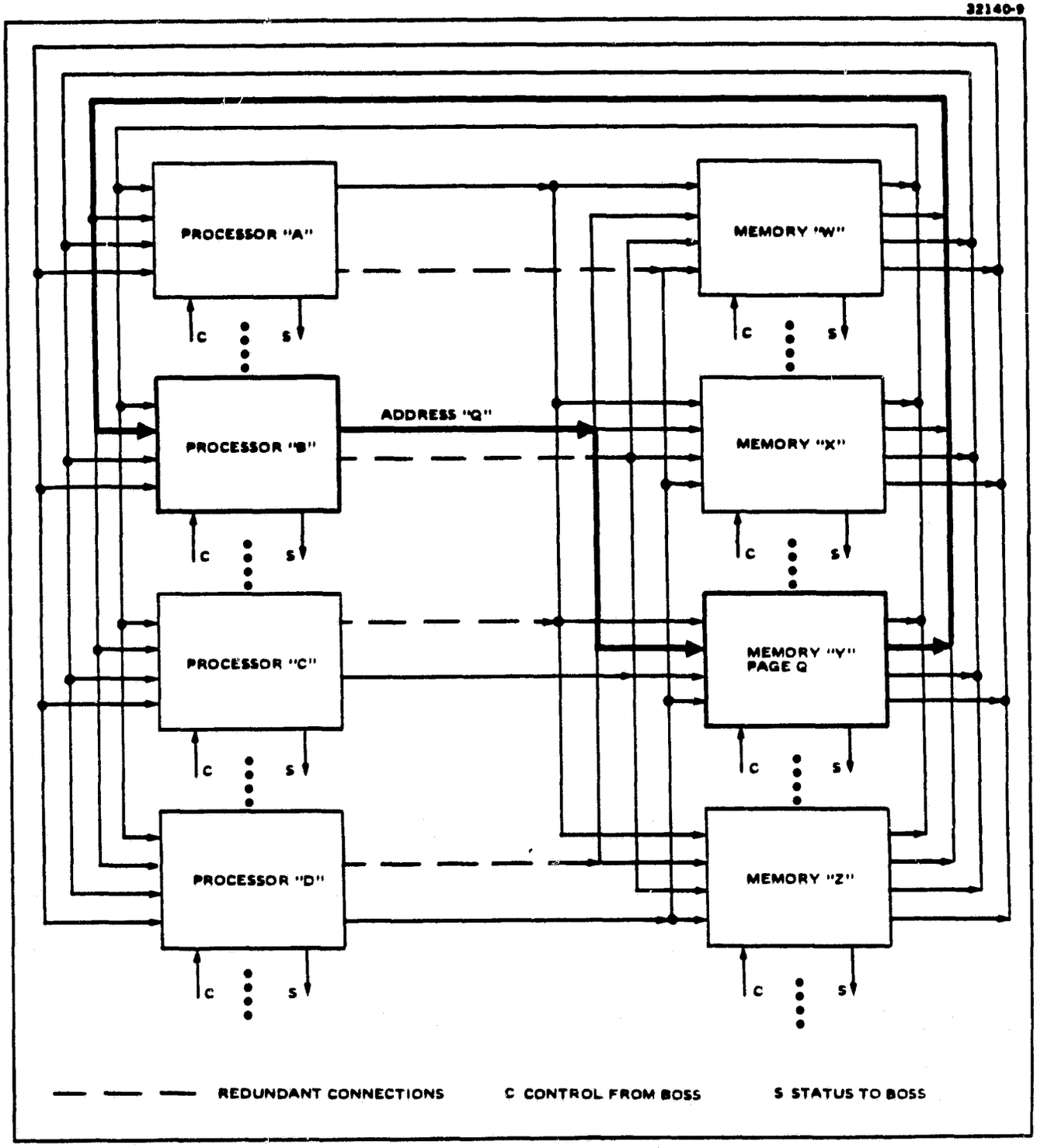


Figure 2. ARMMS Processor/Memory Interconnections – I. Processor B Access to Memory Y In Simplex

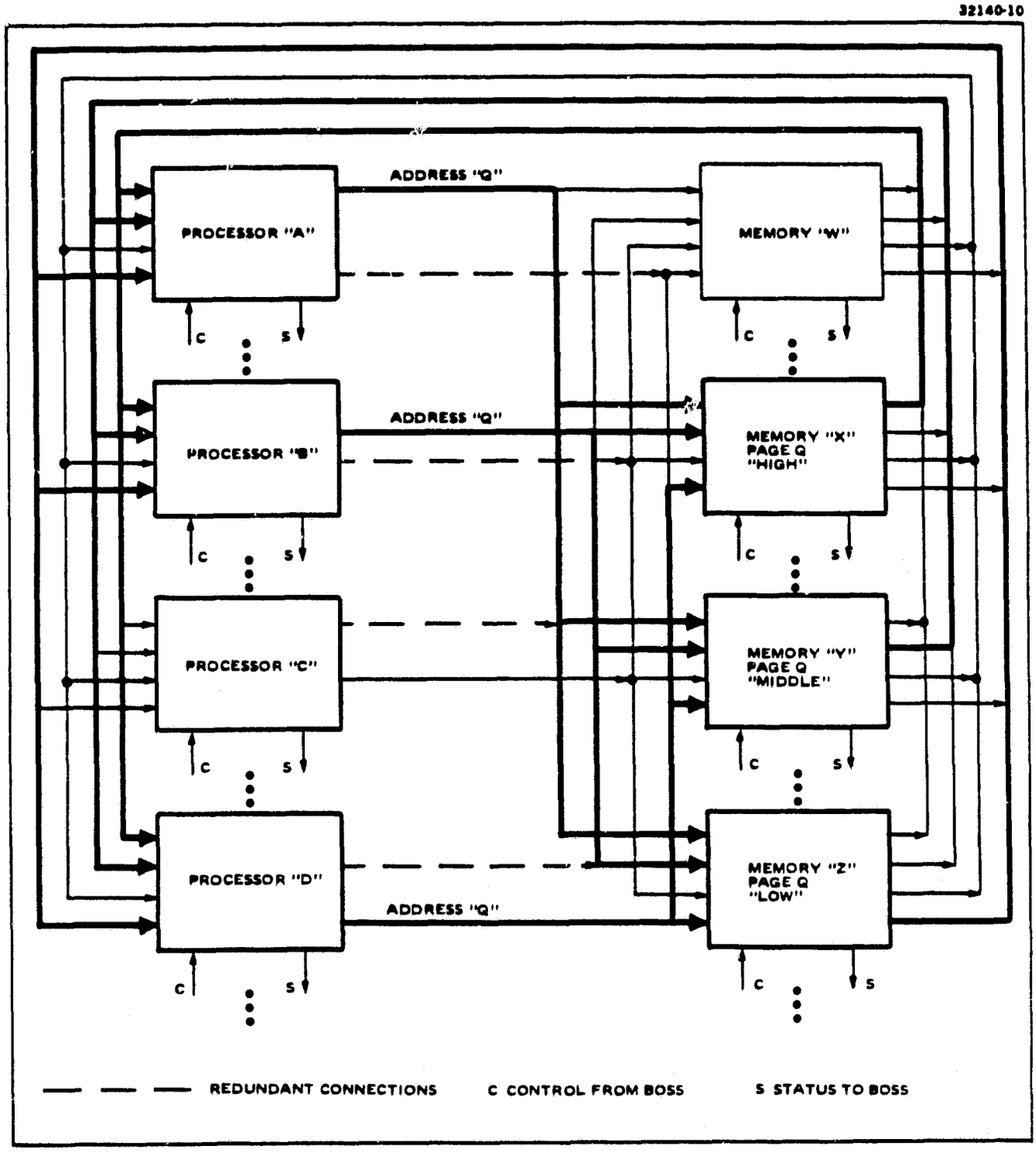


Figure 3. ARMMS Processor/Memory Interconnections – II. Processors A, B, D Access To Memories X, Y, Z X, Y, Z In TMR

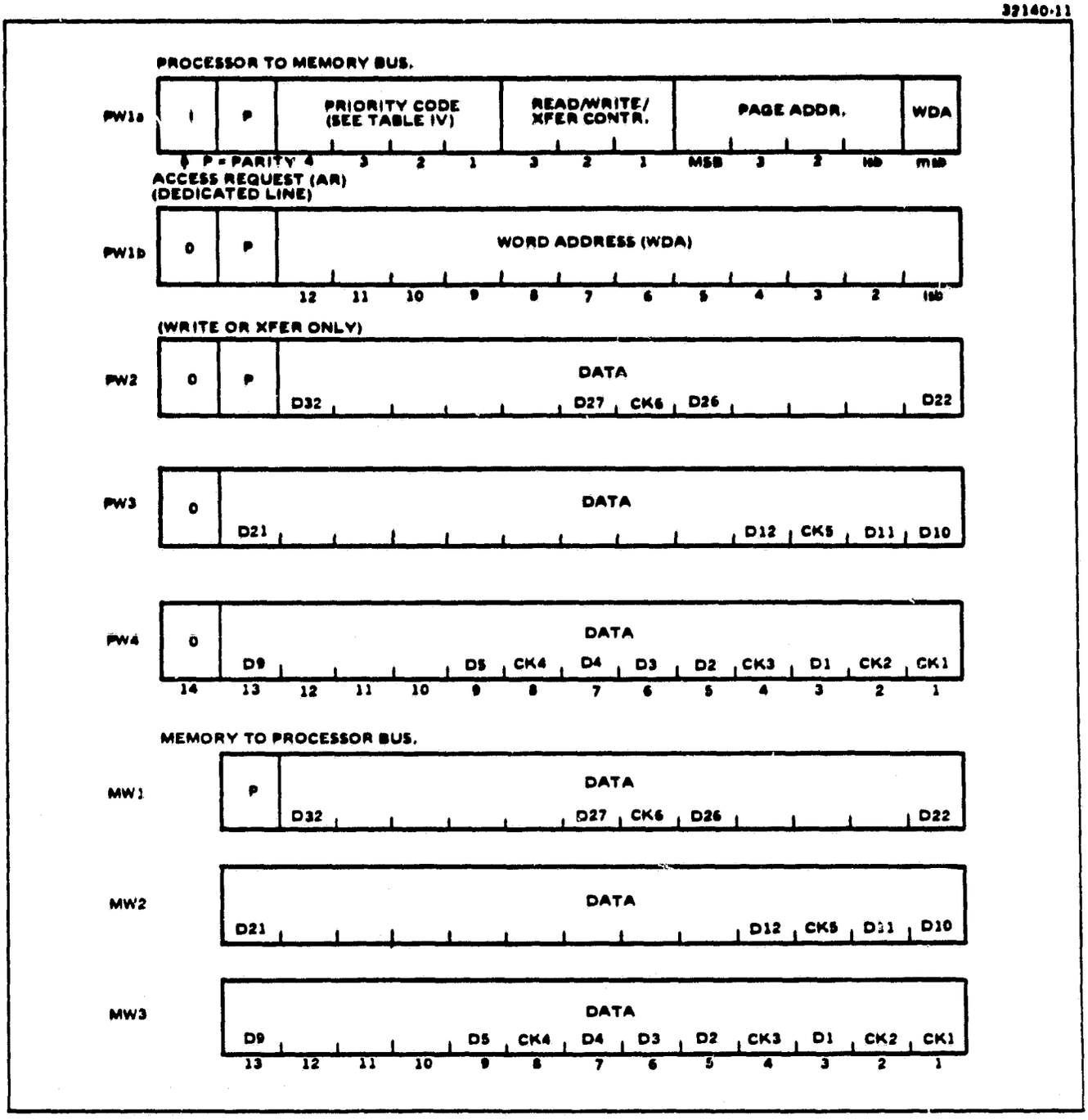
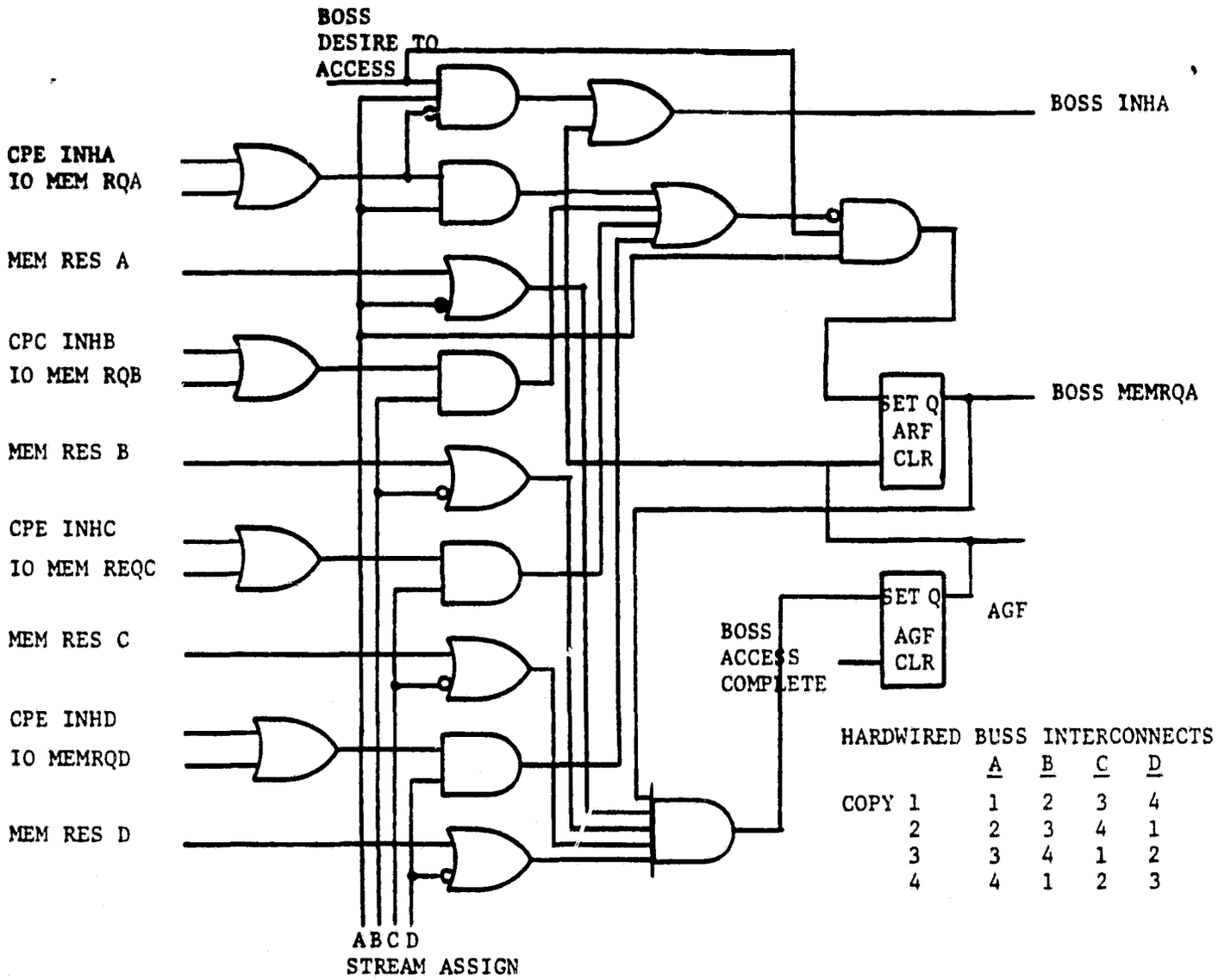


FIGURE 4. ARMMS MEMORY/PROCESSOR WORD FORMATS

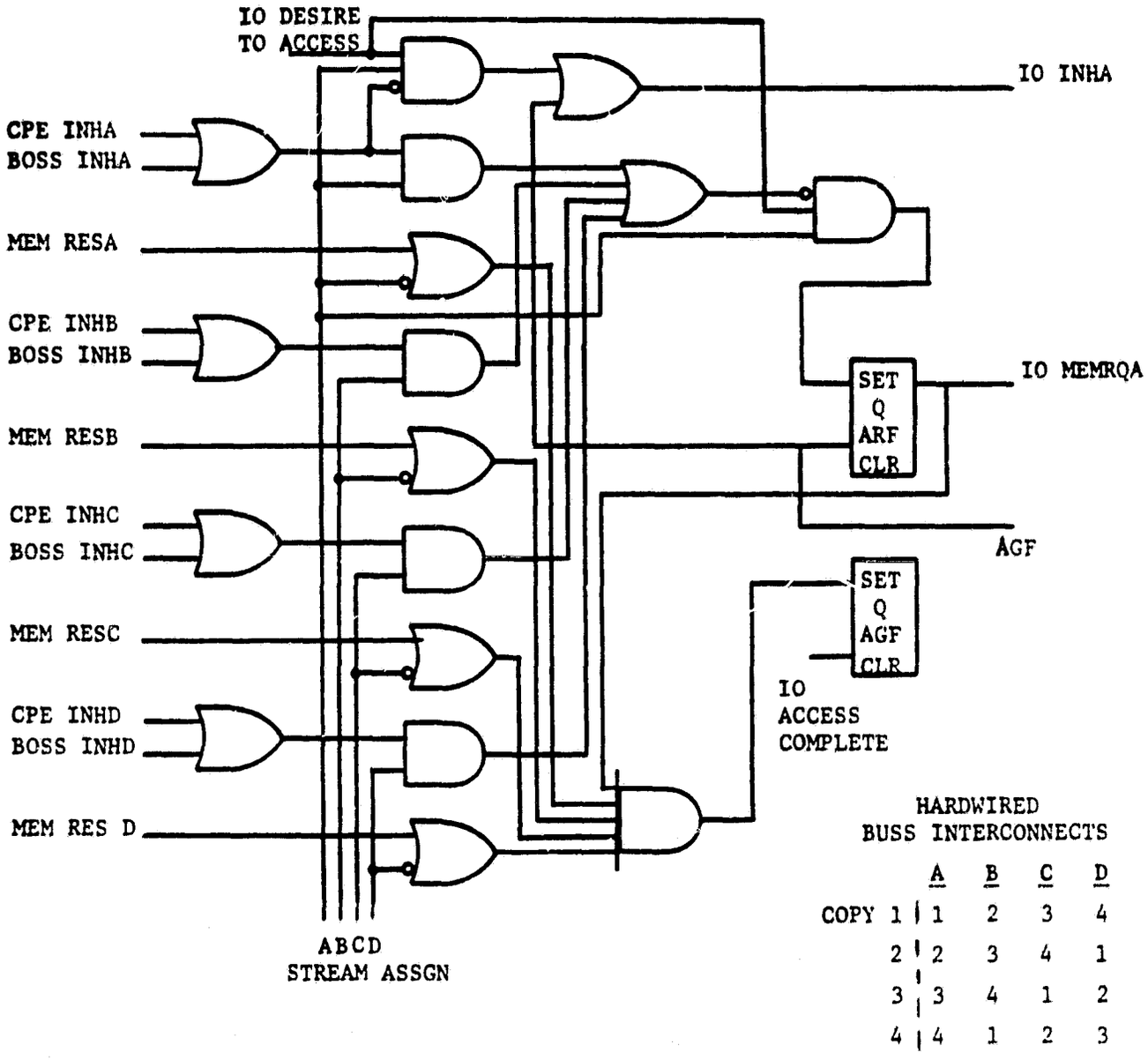


HARDWIRED BUSS INTERCONNECTS

	A	B	C	D
COPY 1	1	2	3	4
2	2	3	4	1
3	3	4	1	2
4	4	1	2	3

FIGURE 6. BOSS MODULE MEMORY ACCESS CONTROL LOGIC

COMPLEXITY 17 GATE, 2 FLIP-FLOP, ~ 20 PINS



HARDWIRED
BUSS INTERCONNECTS

	A	B	C	D
COPY 1	1	2	3	4
2	2	3	4	1
3	3	4	1	2
4	4	1	2	3

FIGURE 7 I-O MODULE MEMORY ACCESS CONTROL LOGIC

Complexity, 17 gates, 2 flip-flop, ~20 pins

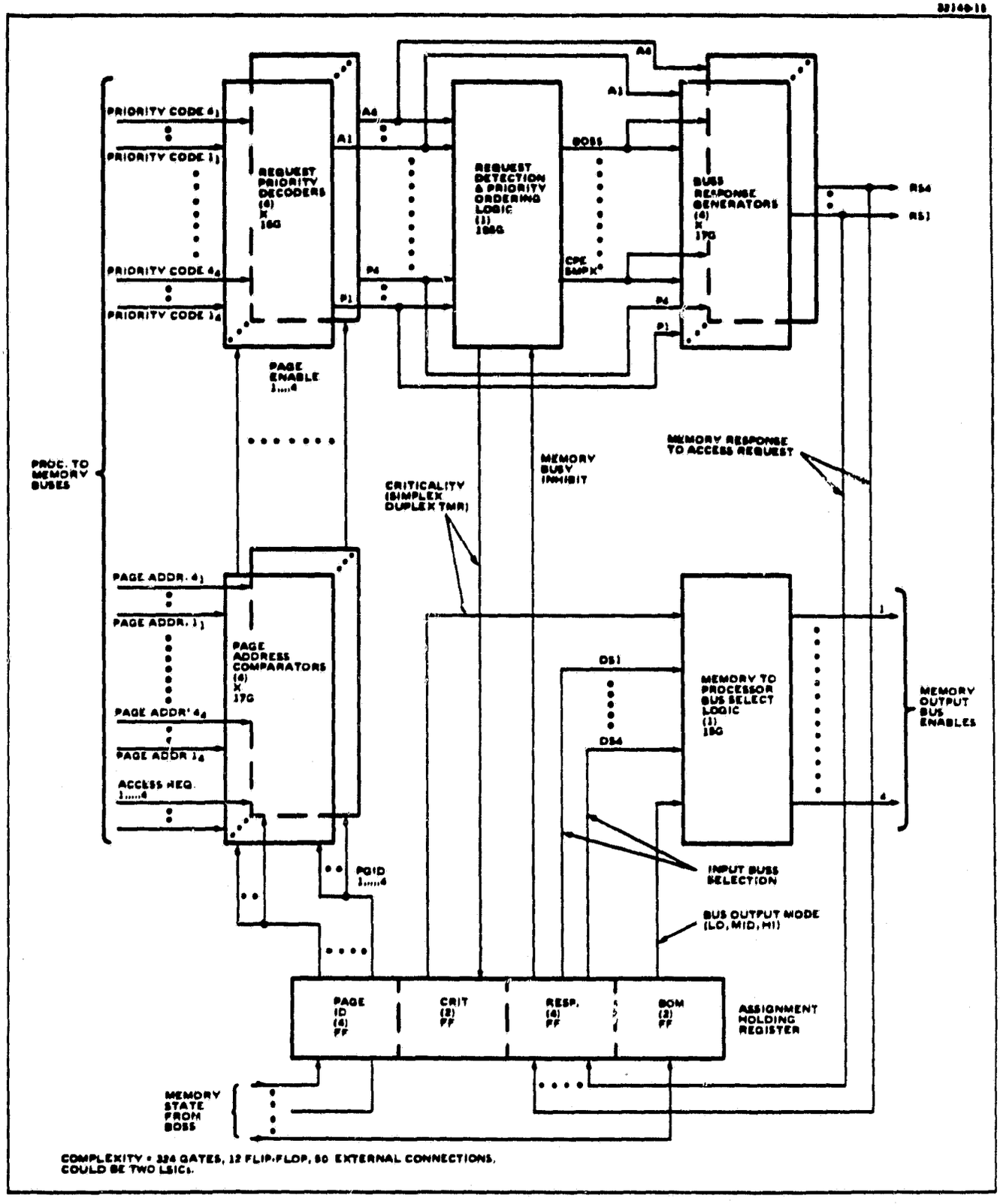


Figure 8. Memory Access Control Logic - (16 Priority Levels)

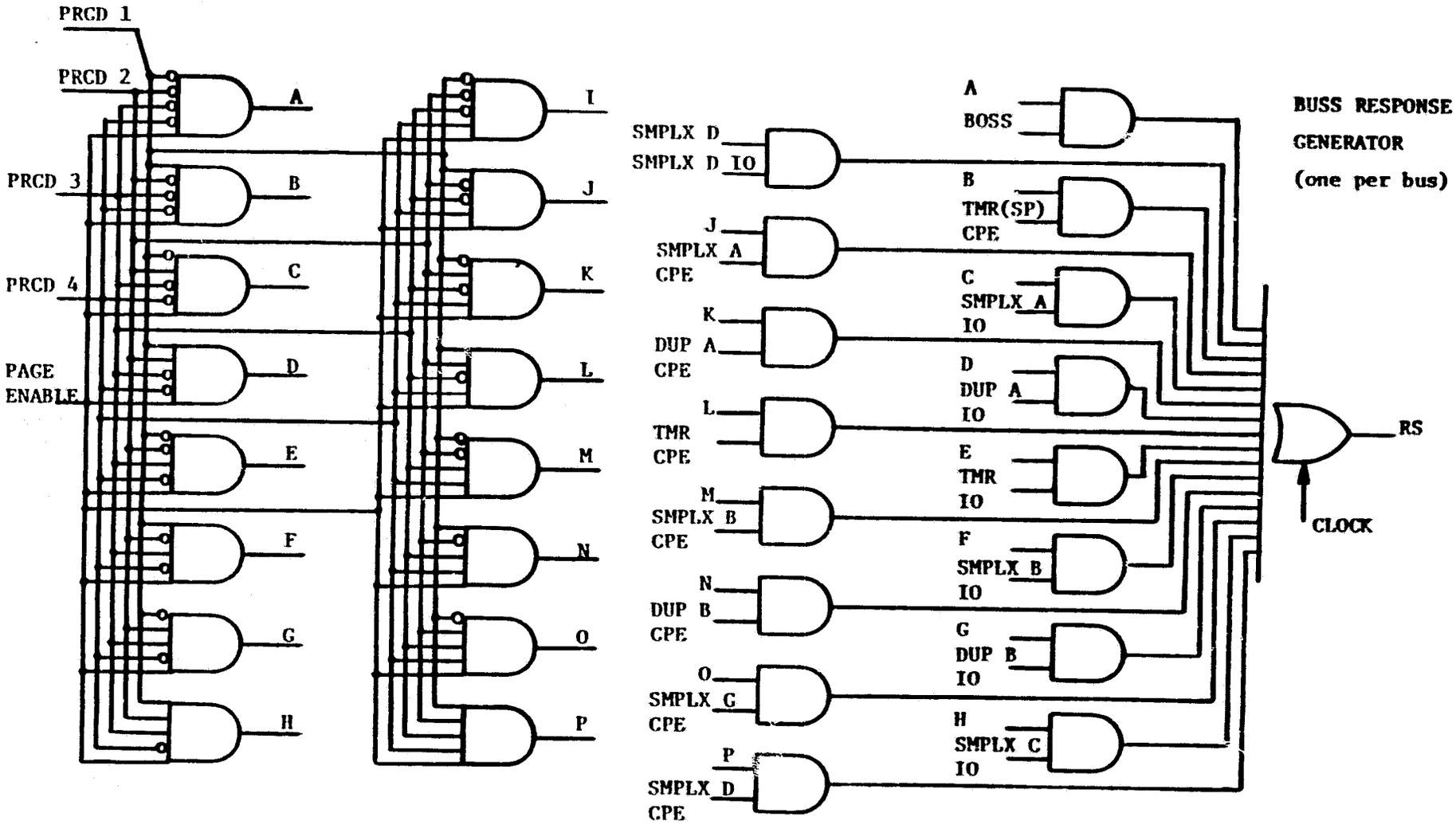
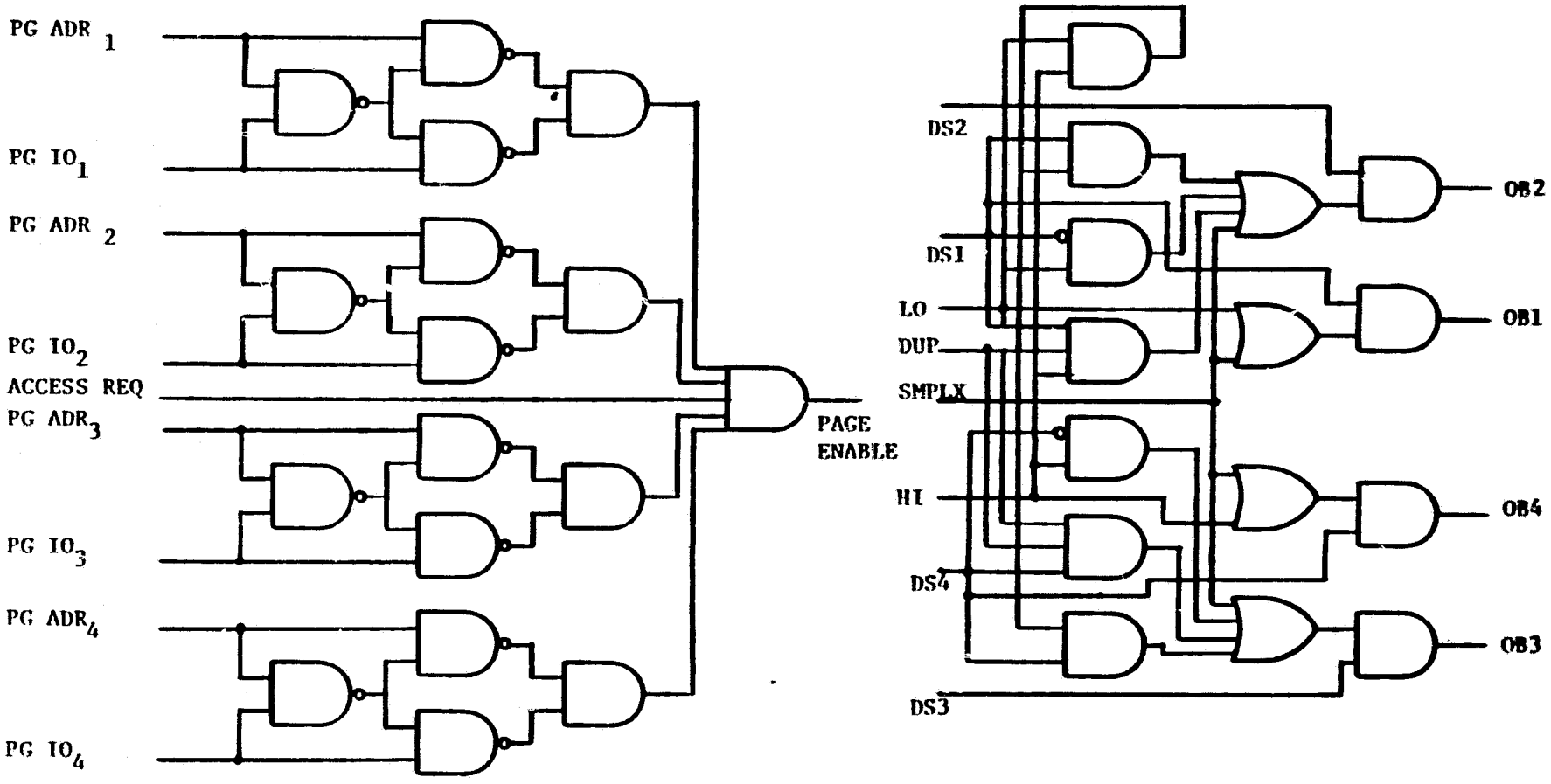


FIGURE 9. MEMORY MODULE ACCESS CONTROL LOGIC - DETAIL I

REQUEST PRIORITY DECODER (one per bus)

PAGE ADDRESS COMPARATOR
(ONER PER BUS)

MEM TO PROC. BUS SELECT LOGIC
(ONE PER MODULE)



PG ADR = PAGE ADDR.

IN FIGURE 6

PR CD = PRIORITY CODE

FIGURE 9. MEMORY MODULE ACCESS CONTROL LOGIC - DETAIL I

REQUEST PRIORITY DECODER (one per bus)

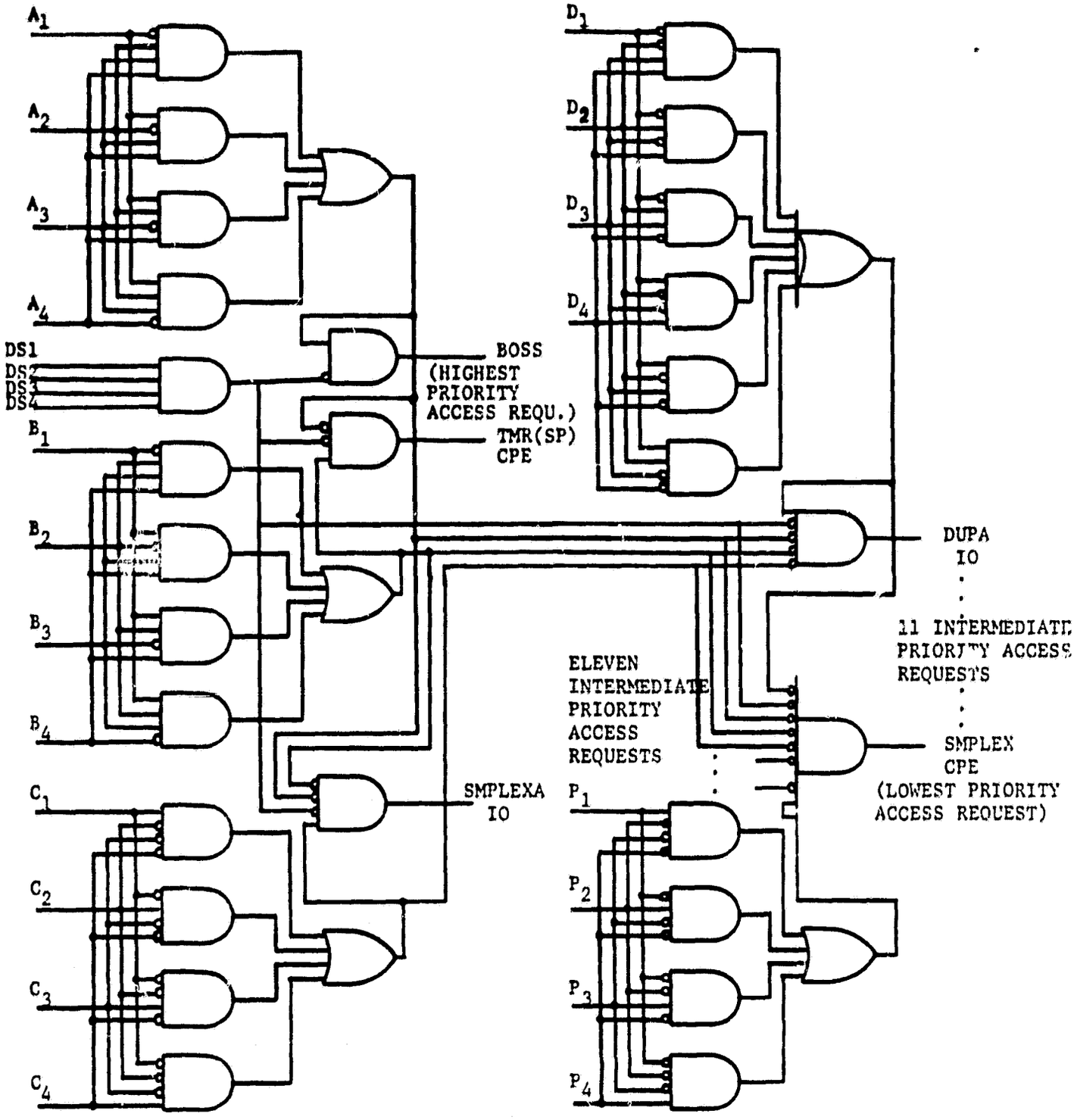


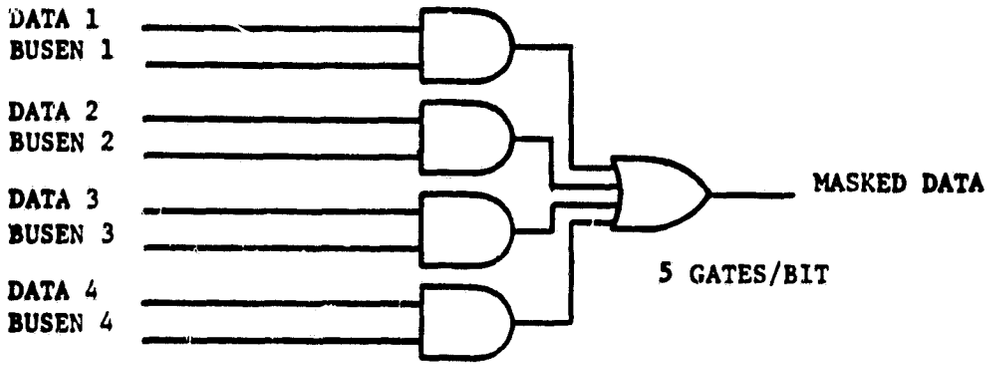
FIGURE 10 MEMORY MODULE ACCESS CONTROL LOGIC - DETAIL II

REQUEST DETECTION AND PRIORITY ORDERING LOGIC (ONE PER MODULE)

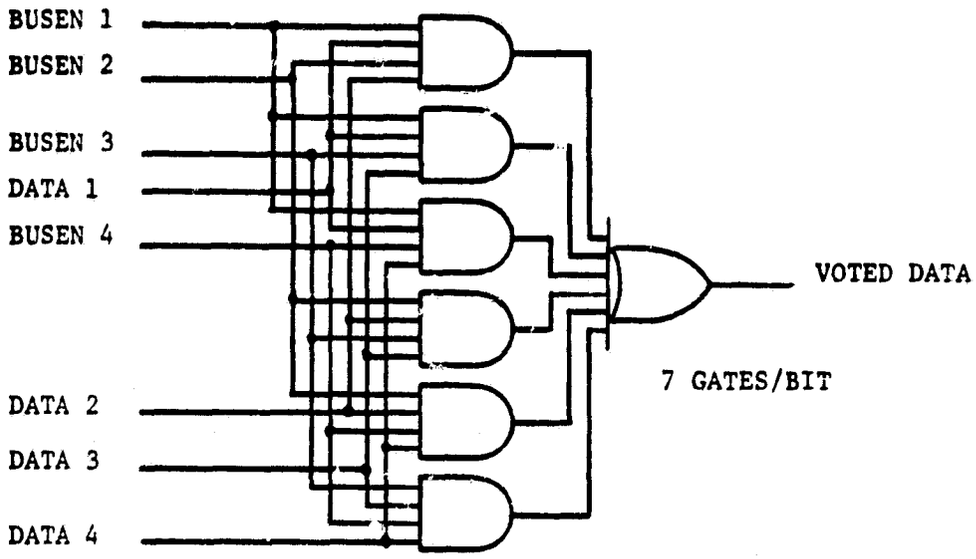
NOTES:

1. ALL SMPLEX DECODING IS DONE AS IN "SMPLEX IO", ALL DUPLEX DECODING AS IN "DUPA FU", AND ALL TMR DECODING AS IN "TMR(SP)CPE".
2. ANY GIVEN PRIORITY LEVEL RECEIVES INHIBITS FROM ALL HIGHER PRIORITY LEVELS.
3. SUBSCRIPTS REFER TO BUSES - I.O. C_4 IS SIGNAL C FOR BUS #4, ETC.

(1) MASKING/SWITCH ONLY (SIMPLEX, DUPLES, TMR)



(2) VOTING ONLY (TMR)



(3) DETECTION ONLY (TMR - NO ISOLATION IS AN INDIVIDUAL BUS)

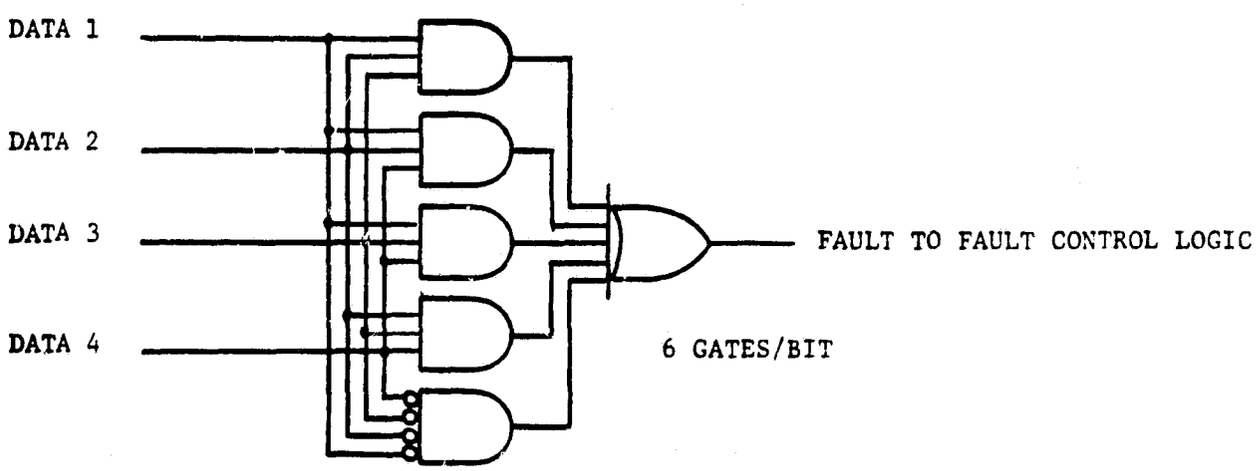


Figure 11 MODULE INTERFACE VOTING, MASKING & ERROR DETECTION LOGIC

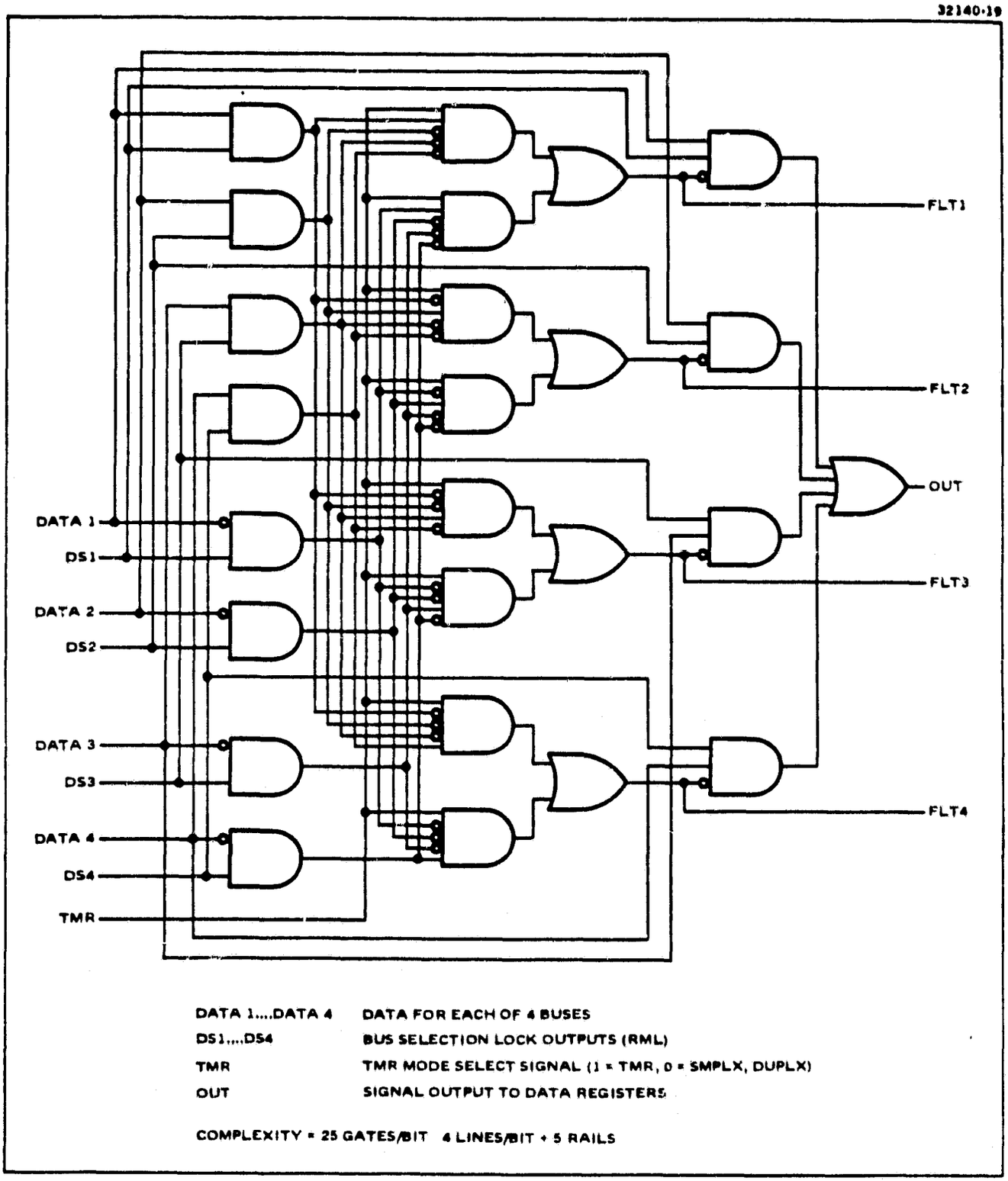


Figure 12. Universal Bus Voter/Switch (one Bit Slice - 13 Required Per Module)

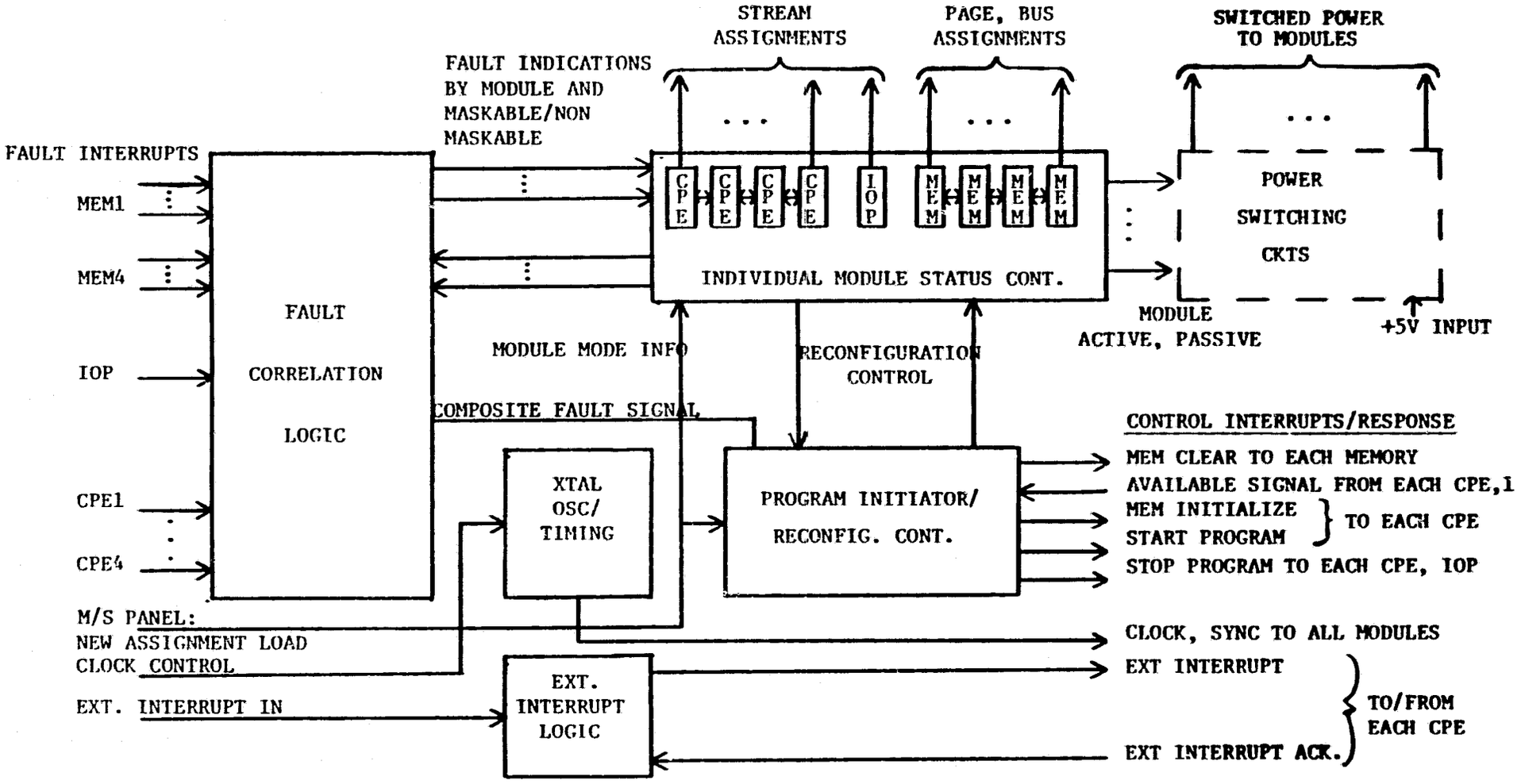


FIGURE 13. ARMS CCE BLOCK DIAGRAM

NOTE: POWER APPLIES TO MODULE UNLESS CONTROLLER IN SPARE OR FAILED STATE, MASTER CLEAR FORCES MSC TO SPARE.

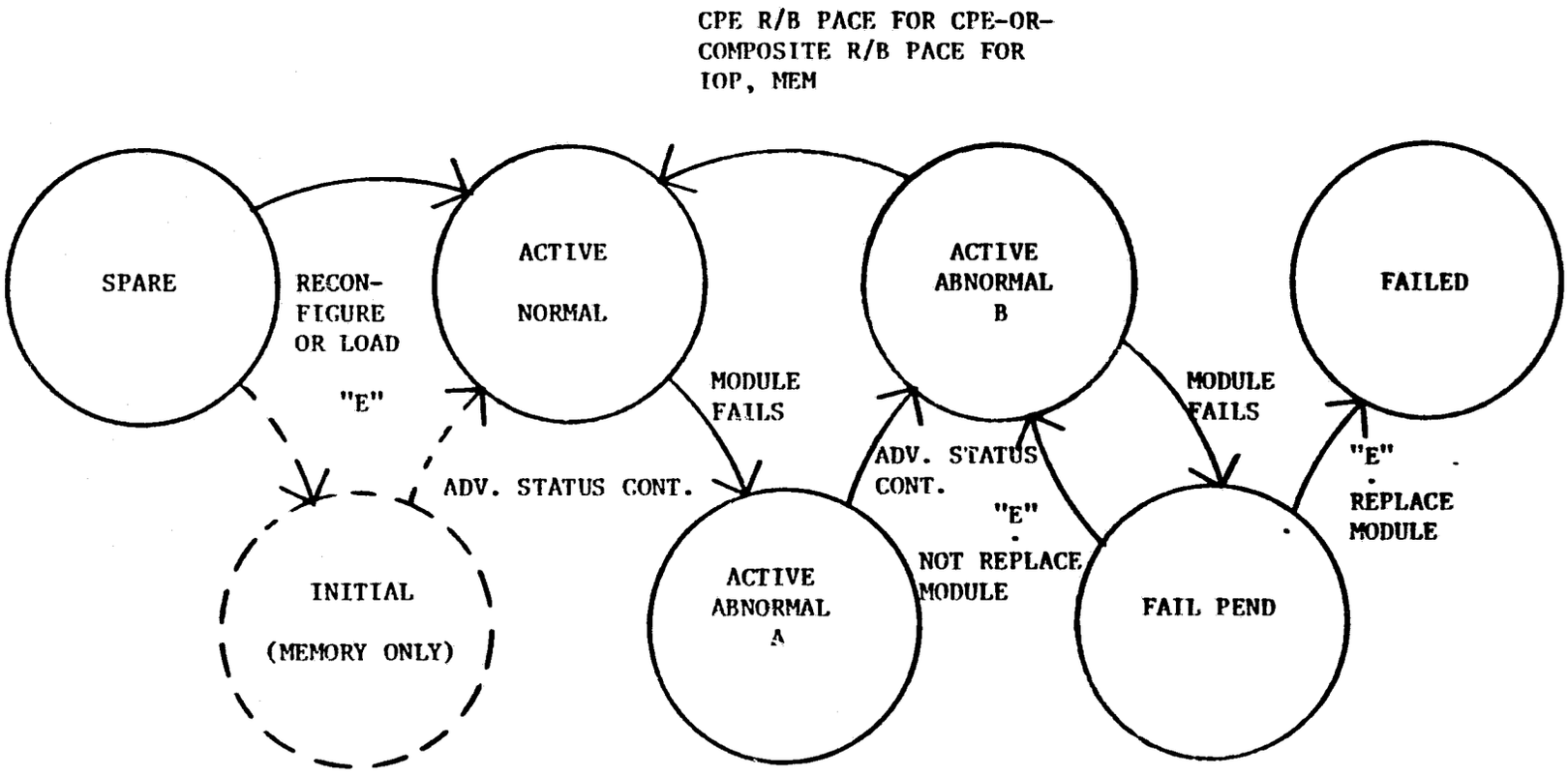


FIGURE 14. CPE MODULE STATUS CONTROLLER STATE TRANSITION DIAGRAM

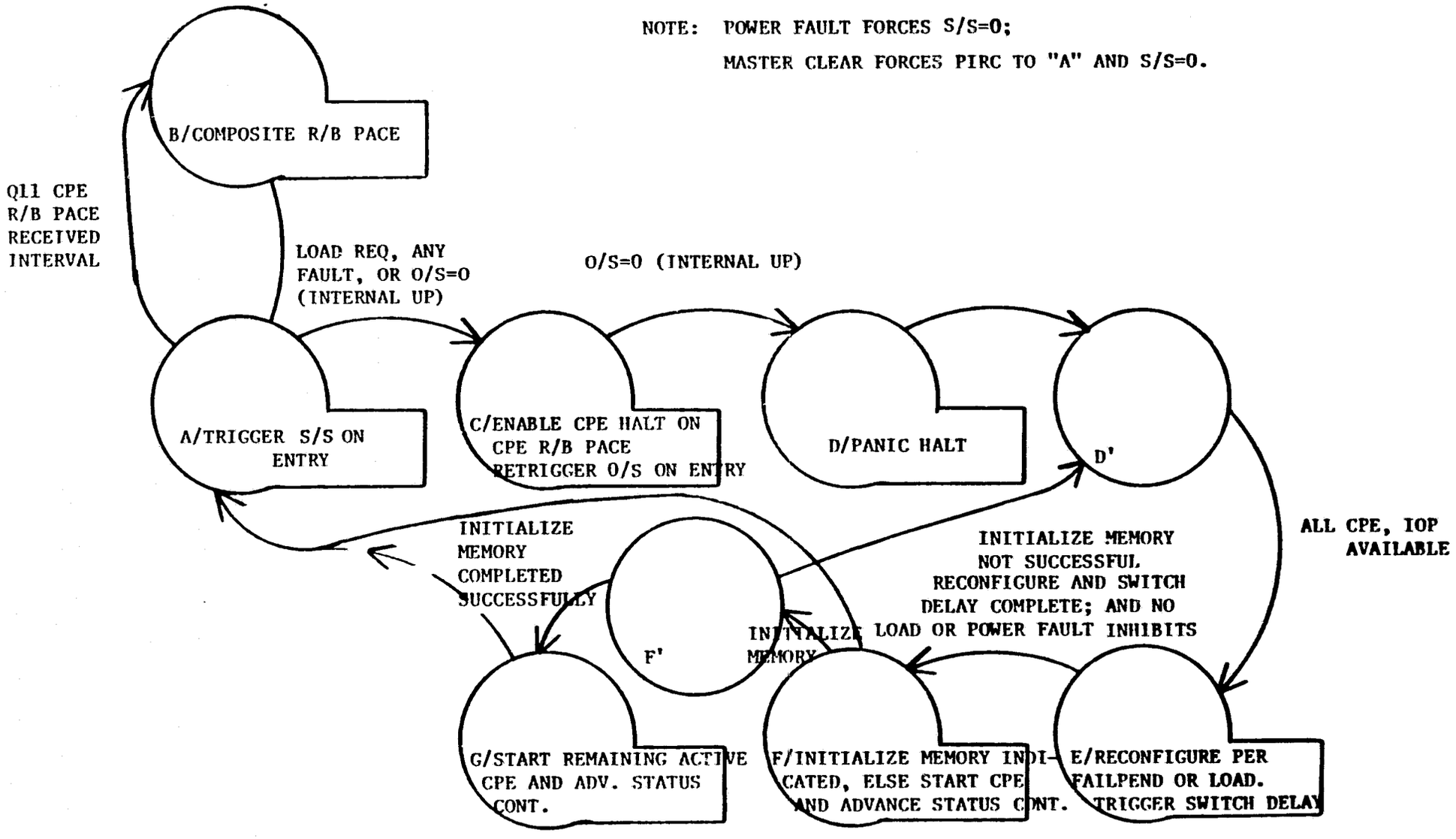


FIGURE 15. PROGRAM INITIATOR AND RECONFIGURATION CONTROLLER STATE TRANSITION DIAGRAM

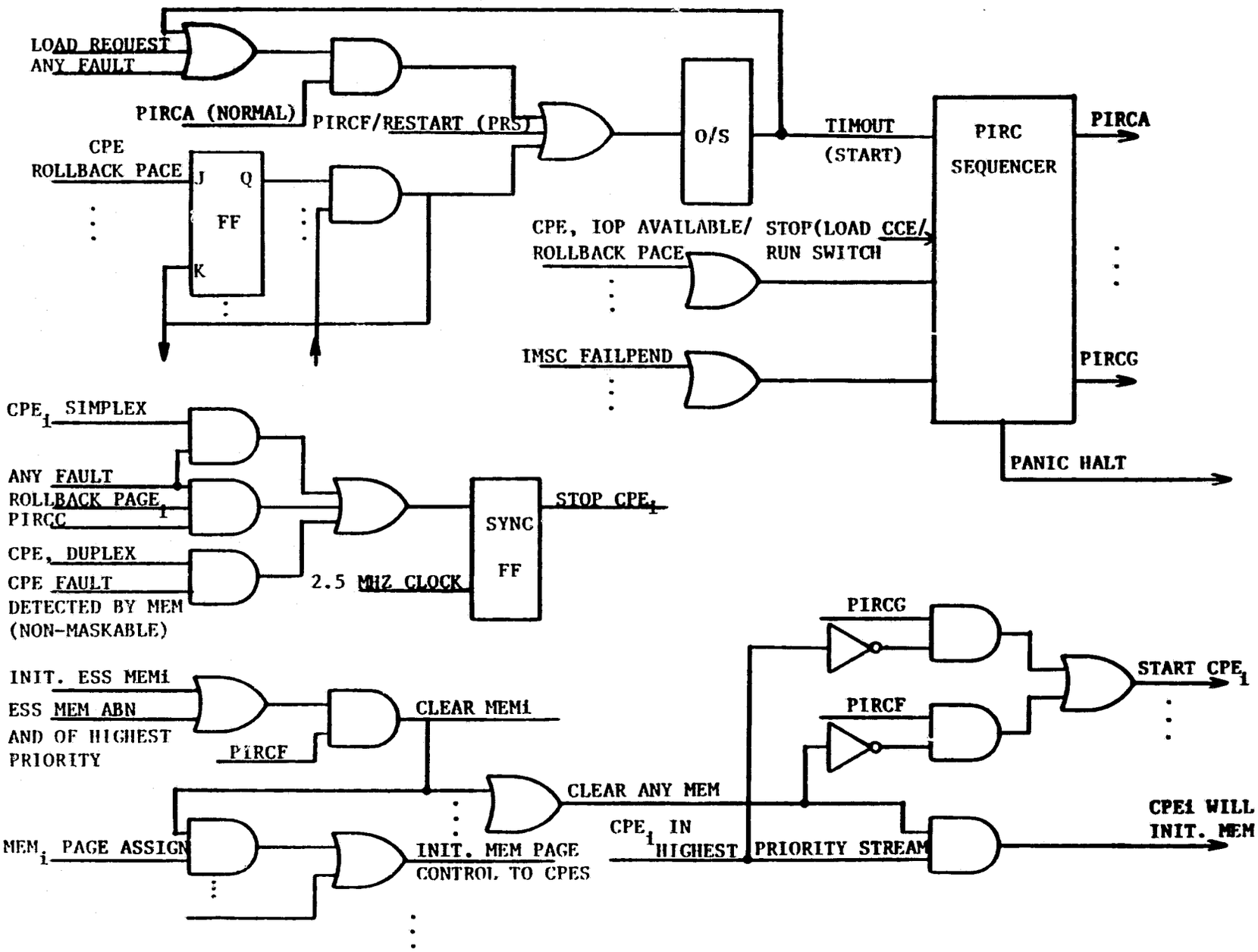


FIGURE 16: CCE PROGRAM INITIATOR/RECONFIGURATION CONTROL (PIRC) LOGIC

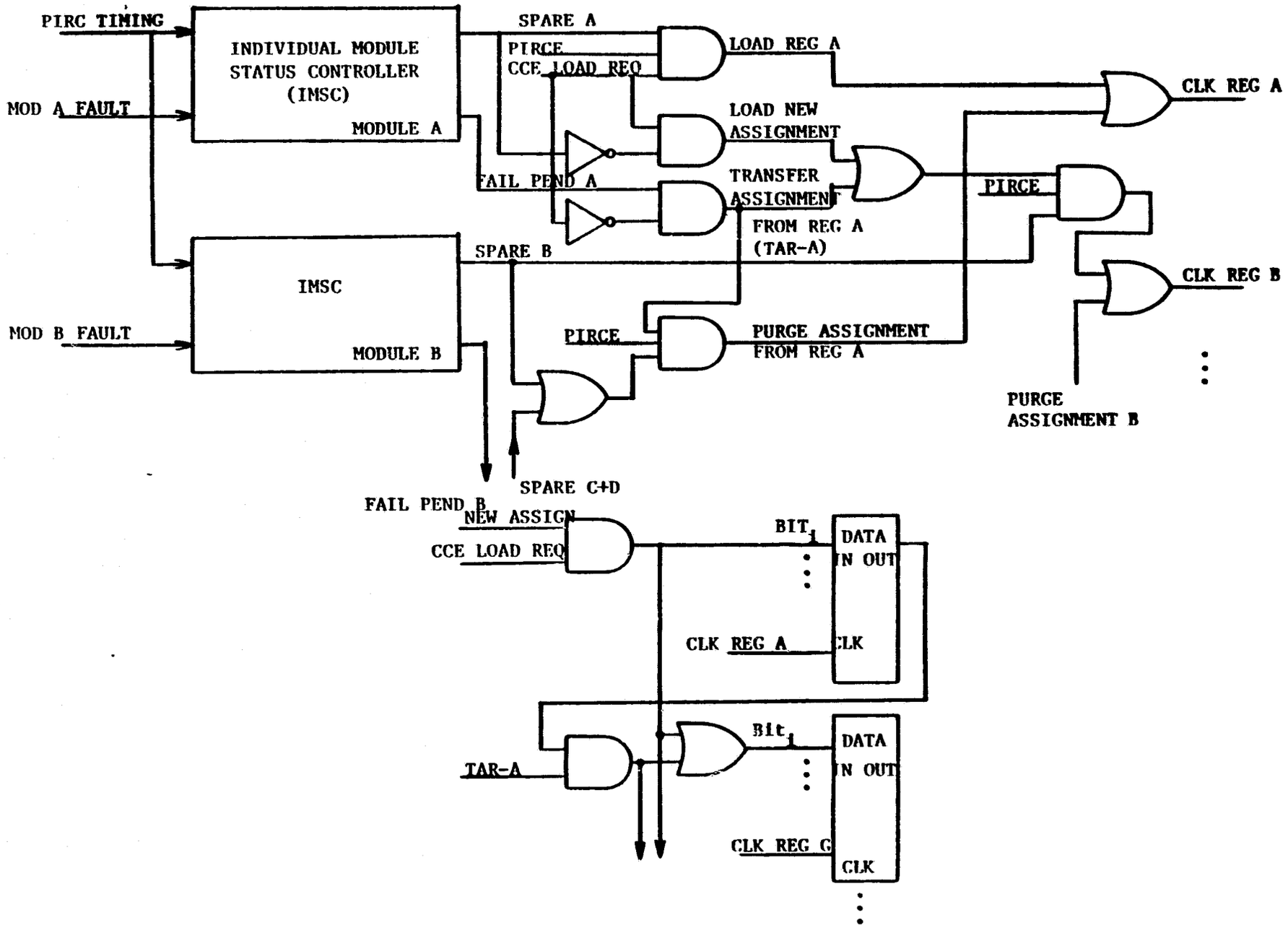


FIGURE 17. CCE MODULE STATUS CONTROLLER LOGIC

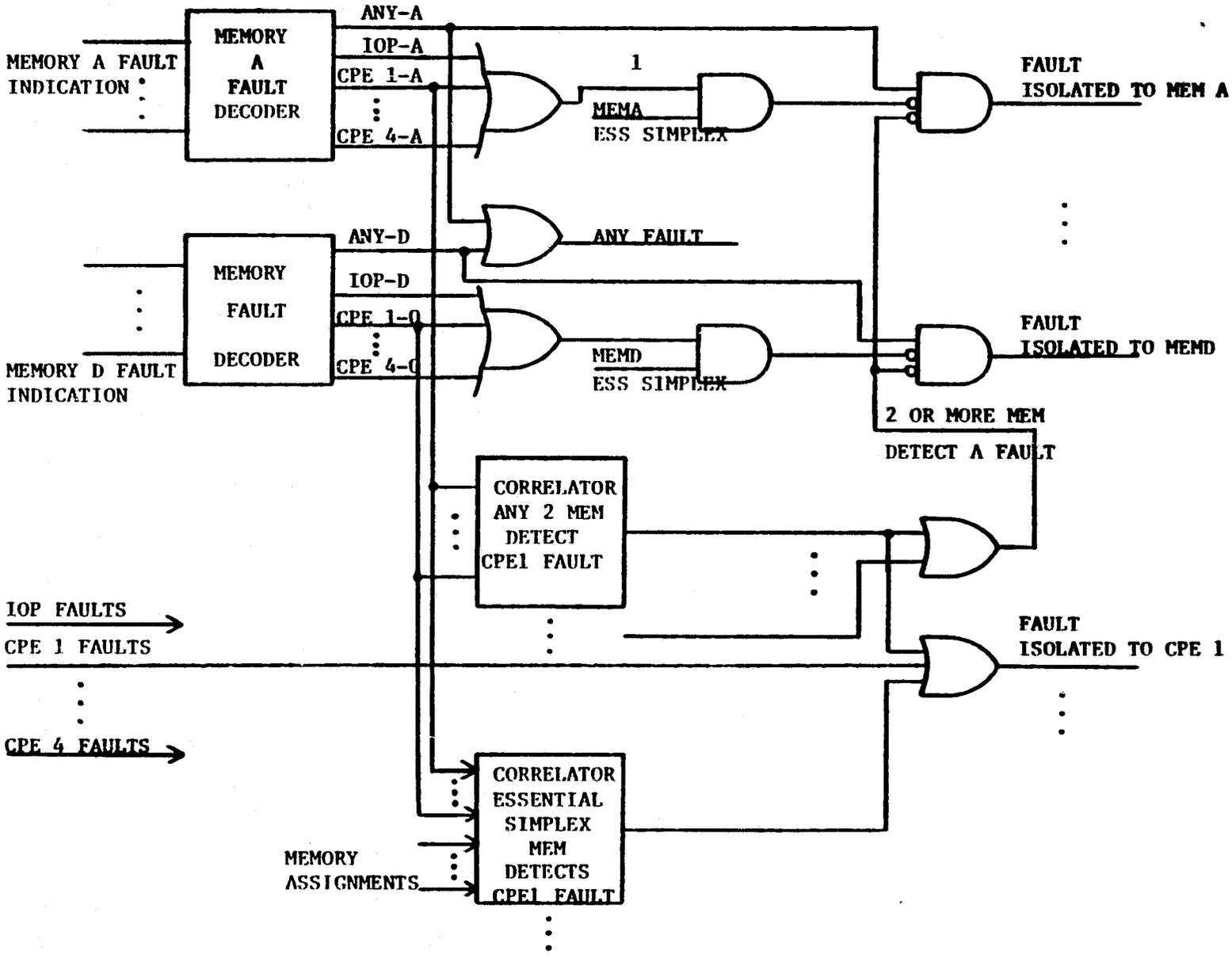


FIGURE 18. CCE FAULT CORRELATION LOGIC

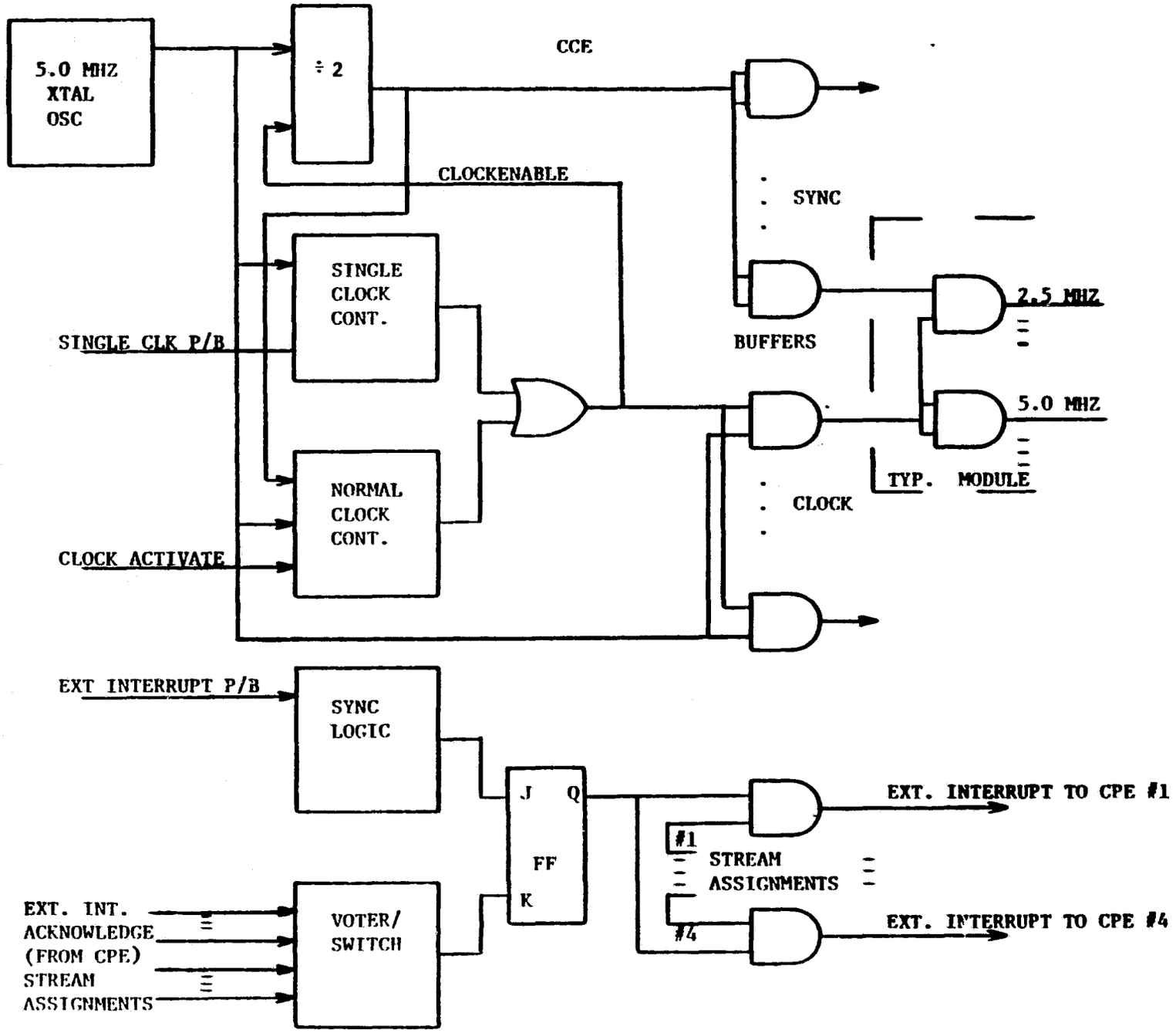


FIGURE 19 CCE CLOCK DISTRIBUTION AND EXTERNAL INTERRUPT LOGIC

DESIGN OF A MODULAR DIGITAL COMPUTER SYSTEM

DRL 5

ARMS ENGINEERING BREADBOARD

FABRICATION AND TEST REPORT

APRIL 25, 1980

Prepared under Contract NAS8-27926

by

HUGHES AIRCRAFT COMPANY

FULLERTON, CALIFORNIA

for

GEORGE C. MARSHALL SPACE FLIGHT CENTER

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

FOREWORD

This report documents fabrication and test of the ARMS Engineering Breadboard accomplished during the fabrication and test phase of contract NAS8-27926 from June 1975 through December 1979. This effort was a follow-on to the architecture study and logic design phases of this contract previously completed and documented.

CONTENTS

I - SINGLE STRING

1. Partial Single String Fabrication

A Partial Single String ARMS Engineering Breadboard (EB), as dictated by incremental funding, was fabricated. The partial string ARMS EB consisted of the following:

1 - Memory Module (MEM)

1 - Central Processing Element (CPE)

1 - Central Control Element (CCE)

1 - Maintenance/Status Panel and Electronics (MSPE)

These modules were assembled (IC's installed, etc.) on subassemblies of the frames that would be installed in the cabinet at a later date. Computer generated wiring programs were utilized to interconnect the IC's with termi-point wiring and to also specify the subassembly interconnections. The modules were housed in a temporary test fixture for the duration of the single string test.

2. SINGLE STRING TEST

The purpose of testing in a single string configuration was to ensure logical and functional correctness of each module before the memory and CPE modules were replicated. A partial single string test was also compatible with funding limitations.

Testing commenced with verification of power distribution thruout the single string and verification of panel functions necessary for CCE testing. The entire CCE module and remaining panel functions were then tested in minute detail so that the CCE would function for single string testing and later for full system testing. Detail test progressed as follows:

- o Clock distribution internal to CCE and distribution to the interface wiring for a complete system.
- o Initialization operation of the module status control logic was verified including module status logic for a complete system. Stream page and buss assignments to the complete system were verified.
- o Operation of the Program Initiator and Reconfiguration Controller was verified including all interrupt/response signals to the complete system.
- o Operation of the Fault Correlation Logic was verified. Each fault interrupt was simulated and the proper response verified.

The CPE module was tested in minute detail for the reasons discussed above and also so that it could be used as a reference later when other CPE's were brought on line and data compared at redundant memory interfaces. Detailed test progressed as follows:

- o Scanout Verification
- o Master clear verification
- o Micro program control operation
- o Registers operation
- o ALU operation
- o Detail verification of each instruction in the instruction set. Various short programs and other methods of inserting data were used to exercise the various paths, options, etc. thru the microcode for each instruction. ROM simulators were used in place of the PROMs so that the stored microcode could be readily changed.

The Memory Module was tested in minute detail for the same reasons as the CPE Module discussed above. Detailed test progressed as follows:

- o Timing & control operation
- o Integration with core memory module
- o Voter switch and output multiplexer logic verification
- o Fault detection logic verification
- o Hamming/Parity encoder and corrector verification

As a demonstration of the fault tolerant capability, a successful, continuous read/write operation was executed with the core memory module logic partially disabled.

3. IOP FABRICATION AND TEST

An IOP module was fabricated and added to the single string. The IOP module was assembled on a subassembly of the same type as the other single

string modules and installed in the temporary test fixture. Computer generated wiring programs were used to automatically wire the IC's. Wire wrap wiring was the most cost effective method of wiring at this point in time.

The IOP Module was tested in minute detail. Detailed test progressed as follows:

- Common Control operation was verified. Handshaking with reference to CAW, CCW, CSW, CC, & IO interrupt was tested.
- TTY channel operation was verified along with the Data Terminal Controller operation. Data transfers and IO instructions were executed to verify the TTY interface.
- Fault Detection logic was verified.

4. SINGLE STRING OPERATION

Single string operation was verified by loading the TTY cassette with a short program, transferring that program from the cassette to computer memory and from memory out to the TTY printer. The program execution verified all IO instructions and other instructions of the SUMC subset.

COMPLETE SYSTEM BUILDUP

1. CPE and Memory Modules Replicated

PROMS for the three additional CPE modules were blown from the updated control prom data. Updated computer generated wiring programs were used to automatically wire the three additional modules of each type, CPE and Memory. The wire wrap method of wiring was used. The additional modules were assembled on the same type of subassemblies as the single string modules.

2. Modules Installed In Cabinet

The replicated modules along with the original single string modules and three additional core memory modules were installed in the ARMS EB cabinet. The backplane was wired interconnecting all modules. Power was connected to cabinet and power distribution tested.

3. Memory Modules Tested

All three memory modules were tested in a like fashion. The internal fault detection logic was utilized to detect fabrication errors (misplaced IC's, wiring errors, etc.). Short programs such as the IOP test program were run and each successive module automatically compared in duplex mode against the previous module at redundant memory interfaces.

The three additional core memories were integrated with memory modules.

The programs run also verified operation of all memory modules with the IOP.

4. CPE Modules Tested

The approach to CPE module testing was almost identical to that of memory module testing. All three CPE modules were tested in a like fashion. The internal fault detection logic was utilized to detect fabrication errors (misplaced IC's, wiring errors, etc.). Short programs such as the IOP test program were run and each successive module automatically compared in duplex mode against the previous module at redundant memory interfaces.

5. Duplex and TMR Operation

Duplex and TMR operations were verified by setting up in the appropriate configuration and running a short program that input the program from the TTY cassette to computer memory, massaged some of the data and output the program to the TTY printer.

6. Reconfiguration

Errors were inserted at the CPE and reconfiguration verified while single clocking thru the operation.

Dynamic reconfiguration was observed at the panel scanouts when errors of opportunity occurred.

7. System Verification Remaining

More exhaustive verification of the ARMS capabilities could be accomplished by injecting a much larger quantity and more varied range of faults. This fault injecting would be particularly effective if accompanied by a more thorough diagnostics program.

8. Problems Encountered

No problems of a system concept nature were encountered. The detailed logic tests and operational tests indicated that the system performed as conceived.

An area of checkout where many design problems were resolved was the CPE microcode debug. Resolution of these problems was relatively easy because the microcode was stored in ROM simulators which facilitated correcting the code.